NISTIR 90-4259

SECURE DATA NETWORK SYSTEM (SDNS) ACCESS CONTROL DOCUMENTS

Charles Dinkel
Editor

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Computer Systems Laboratory
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary

Lee Mercer, Deputy Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

NIST

NISTIR 90-4259

SECURE DATA NETWORK SYSTEM (SDNS) ACCESS CONTROL DOCUMENTS

Charles Dinkel
Editor

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Computer Systems Laboratory
Gaithersburg, MD 20899

February 1990

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
Lee Mercer, Deputy Under Secretary for Technology
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

Table of Contents

FOREWORD

The Secure Data Network System (SDNS) architecture and a set of associated specifications were developed through a multi-organizational project sponsored by the National Security Agency (NSA). They are presented here as a basis for standardization of security services in the Open Systems Interconnection (OSI) architecture. The National Institute of Standards and Technology (NIST) intends to encourage widespread adoption of the resulting standards and the implementation of these security services into a wide spectrum of vendor products.

NIST is publishing the specifications that resulted from Phase I of the SDNS project for review and comment from potential government and commercial users of security products. The specifications are not complete or totally consistent, either internally or with a number of other security projects in the National and International Standards arena. Readers of these documents should recognize that these specifications are subject to modification for various reasons as they progress through the standards process. The sponsor and participants in the SDNS project are acknowledged for the work accomplished and their support in developing and releasing these specifications.

The SDNS project was initiated by NSA to investigate methods of implementing security in a distributed computer network. The results of this project include a set of specifications that include security services, protocols and mechanisms for protecting user data in networks that are based on the OSI computer network model. Productive security services that protect user data are specified and supportive security services, such as key management and access control, are also provided. No cryptographic algorithms are included in these specifications.

NIST is working with NSA and industry to identify and develop a framework of base standards for network security. In 1989, NIST established the OSI Security Laboratory where interested researchers from government and industry develop and demonstrate new ideas in network security. The major goals of NIST's network security activities are to:

- Identify and develop security standards for open systems
- Specify a key management system that supports these security standards
- Encourage the development of interoperable equipment

The documents resulting from Phase I of the SDNS project are as follows:

SDN.301 - Security Protocol 3 (SP3)
SDN.401 - Security Protocol 4 (SP4)
SDN.601 - Key Management Profile - Communication Protocol Requirements for Support of the SDNS Key Management Protocol
SDN.701 - Message Security Protocol
SDN.702 - SDNS Directory Specifications for Utilization with the SDNS Message Security Protocol
SDN.801 - Access Control Concepts Document
SDN.802 - Access Control Specification
SDN.902 - Key Management Protocol - Definition of Services Provided by the Key Management Application Service Element
SDN.903 - Key Management Protocol - Specification of the Protocol for Services Provided by the Key Management Application Service Element
SDN.906 - Key Management Protocol - SDNS Traffic Key Attribute Negotiation

Because of the wide spread interest in the SDNS project, NIST is publishing these ten documents as three Reports entitled: **Security Protocols, Key Management,** and **Access Control.** The following diagram shows the relationship and contents of these reports.

NIST REPORT				
SECURITY PROTOCOLS	SDN.301 SECURITY PROTOCOL 3 (SP3)	SDN.401 SECURITY PROTOCOL 4 (SP4)	SDN.701 MESSAGE SECURITY PROTOCOL	SDN.702 DIRECTORY SPECS FOR USE WITH MSP
KEY MANAGEMENT	SDN.601 KEY MANAGEMENT PROFILE	SDN.902 KMP DEFINITION OF SERVICES PROVIDED BY KM ASE	SDN.903 KMP SERVICES PROVIDED BY KM ASE	SDN.906 KMP TRAFFIC KEY ATTRIBUTE NEGOTIATION
ACCESS CONTROL	SDN.801 ACCESS CONTROL CONCEPT DOCUMENT	SDN.802 ACCESS CONTROL SPECIFICATION		

INTRODUCTION

NISTIR 90-4259 consists of three documents developed by the National Security Agency (NSA) as output from the Secure Data Network System (SDNS) project. The **Access Control Concept Document, SDN.801,** describes the principles and functions underlying the SDNS access control and authentication security services. It is a goal that the access control and authentication mechanisms designed by SDNS be adaptable to support a wide range of anticipated customer security policies.

The purpose of **SDN.802**, the **Access Control Specification**, is to provide a common basis from which devices implementing the access control service will be able to achieve interoperability. The document also identifies points of reference for users implementing the SDNS Security Protocols for Network, Transport, or Messaging. **SDN.802** gives a functional description of the SDNS access control system and establishes a point of reference from which security protocols can make use of the access control service.

The **SDN.802** specification also provides an overview of the Access Control Information Specification (ACIS). ACIS provides a uniform method for encoding access control information which is independent of any particular security policy. It also provides a standard algorithm for interpreting and comparing access control attributes.

The third document in this set, **SDN.802/1, ACIS Addendum 1**, is an extension of the ACIS discussion provided in section 5 of **SDN.802**. It furnishes a detailed explanation of the capabilities, limitations, and implementation requirements for ACIS.

The access control documents of **NISTIR 90-4259** support the security protocols addressed in **NISTIR 90-4250** and the key management services covered in **NISTIR 90-4262.**

Comments and feedback are solicited by NIST.

ACCESS CONTROL CONCEPT DOCUMENT (REVISION 1.3)

26 July 1989

Preface

This paper has resulted from the developmental work accomplished within the Secure Data Network System (SDNS). This paper addresses SDNS access control and, as such, represents the consensus of the Access Control Working Group (ACWG). Other SDNS working groups, such as Protocol and Systems Management have addressed the other major components of the SDNS. All of these have had a direct influence on the preliminary SDNS access control concepts that are presented in this paper.

Table of Contents

List of Figures

1. INTRODUCTION: ACCESS CONTROL WITHIN SDNS

1.1 Overview

This document describes the principles and functions underlying the Secure Data Network System (SDNS) access control and authentication security services. A trusted distribution algorithm, operating in conjunction with a trusted central authority, provides a means for an authenticated exchange of identity and attribute data between communicating peers. Auxiliary vectors (AVs) provide a means for local authorities to represent additional identity and attribute data within their jurisdiction, without the central authorities' involvement. Based on these and other inputs, a range of rule-based access control (RBAC) and identity-based access control (IBAC) can be afforded in order to satisfy customer requirements. Access approval, enforcement, and authentication functions are defined in a general fashion, compatible with real time and store-and-forward communication contexts.

With the intention of applying multiple vendor SDNS security products to a wide variety of communicating data systems, the SDNS security architecture does not limit secure communications to security products provided by any one vendor. As a result, the access control and authentication security services provided by any SDNS security product must share a common structure. It is a goal that the access control and authentication mechanisms designed by SDNS be adaptable to support a wide range of anticipated customer security policies. The application and variety of security policies influenced the development of a four-tiered model to represent access control identification information.

A framework has been developed for authentication data and access control checks which will allow communication between different SDNS users/systems when their respective security policies allow it. SDNS access control consists of two processes: the Peer Access Approval process for interpreting the data of the four-tiered model, and the Peer Access Enforcement process for enforcing access control on a Protocol Data Unit (PDU) basis.

Security policies are generally established by different administrative levels within an organization. The policies established at each administrative level should either reflect the consistent implementation of the policy established by the administrative level above it or establish security policies that are unique to the current administrative level. These policies have been broadly categorized as rule based and identity based policies.

In order to implement a security policy within any security product, the security policy specification must translate the security policy into a realizable structure (such as an algorithm) and identify the characteristics of each entity which will serve as inputs to the security policy algorithm. The application of a security policy algorithm to the joint characteristics of each communicating entity, along with establishing the validity of each entity's characteristics, results in an access control decision.

SDNS must support the dynamic mechanisms which identify the communicating entities, mediate their access, and enforce the security policy of that specific communication. Most end system and data access decisions will be made as a machine function on behalf of the SDNS users. Toward that end, the SDNS authentication and access control functions are based on the following five assumptions:

1) Security policies that can be implemented by SDNS are those that are mechanistically specifiable (algorithmically expressable).[1]

1. Not all policies are expressable in a mechanistically implementable form.

2) A set of characteristics of the communicating entities is input to an algorithmic security policy decision process. These characteristics have been subdivided into four subsets which are identified as the four-tiered model.

3) Authentication of the SDNS accountable entities is established through a strongly trusted distribution method. This distribution method must employ nonforgeable identification and authentication. From this basis, peer entity authentication is expanded to meet the requirements of SDNS authentication for access control.

4) SDNS provides two processes for access control decisions. The first process determines whether communications can be initiated, and the second process provides continuing access control of data. These are the Peer Access Approval (PAA) and Peer Access Enforcement (PAE) processes.

5) SDNS cannot correct deficiencies in end systems.

The SDNS access control concept document centers on the above second and fourth bullets. The discussion of these two bullets is couched in terms of a required strong authentication process and results in a specific operational access control concept for SDNS.

1.2 Scope & References

This access control concept document provides a framework for understanding SDNS access control; it does not provide the detail that would allow the implementation of SDNS access control in any particular environment. These concepts will be used to develop the SDNS Access Control System Specification (SDN.802).

1.3 Relationship of Authentication and Access Control Within SDNS

A strong authentication mechanism is required through which an SDNS component receives the information necessary to identify and validate another SDNS component. Each communicating SDNS component determines the identity of the other accountable entity from this strong authentication information. The access permissions, limitations, or constraints enforced during that association are established using the identity information received from a trusted central authority in the form of an identity certificate. Note that the identification information contained within an identity certificate will not be uniform over the entire family of SDNS components, but varies with regard to many factors (i.e., the OSI layer at which the SDNS component operates, local security policy).

1.3.1 Exchange

SDNS access control decisions begin with the accountable entities attempting to communicate. These decisions are made in the context of each accountable entity's security policy, as established by the appropriate cognizant authority. Once initialized, the SDNS components can establish the means to securely communicate. Identity certificates are exchanged on behalf of the SDNS accountable entities. Access control enforcement occurs as a result of using the information contained in the identity certificate and continues as long as the specific association exists. The information contained within the identity certificate is intentionally left as general as possible to help preserve interoperability. This allows authorities to configure a wide variety of security policies to govern access control decisions. The information contains enough granularity to allow for access control decisions according to the security policies applied.

1.3.2 *Authentication*

Accountable entity authentication means that those identity characteristics, attributable to the entity being protected by the SDNS component and meaningful for access control, are exchanged and verified. The chosen authentication scheme must support the goal of confirming (authenticating) the identities of the SDNS components.

The COMPOSITE ID (the total of all identity information about an accountable entity as contained in the identity certificate and any extensions, see Section 1.3.4) carries the information necessary for an SDNS component to make an access control decision commensurate with the security policies involved.

1.3.3 *Access Control*

Two variations of an authentication process are discussed in this document, real time and staged delivery. Both variations provide equivalent outcomes. Either an association between entities is established (and supervised according to each SDNS component's access rules) or communication is denied. The authentication mechanism must establish the identity of each SDNS accountable entity involved in the requested secure communication. Once the identity information exchange process has been accomplished another process will be initiated to determine whether or not the secure communication should be allowed. This process is called the PAA process. The PAA process initially determines the constraints imposed upon the secure communication between peers. These constraints are established by interpreting the identity certificate data of both SDNS components and placing the resulting constraints, limitations, or permissions which apply to information present in a PDU label, within an Enforcement Vector (EV). If a valid EV has been formed, the communication is established and the PAE process monitors each inbound and outbound PDU for conformance with the EV's constraints. The information in the EV and the PDU may be represented differently, thus a translation may be required before an access control decision may be made on a PDU. PDUs that conform pass unimpeded; PDUs that do not conform are dropped.

1.3.4 *Extension to the Identity Certificate*

The identity certificate may not be large enough to contain all of the information which is needed to adequately identify an SDNS user/component, and the lifetime of some of the access control information may be too short to be included in the identity certificate. Therefore, a scheme was devised to place all of the locally defined information under the control of a local authority. Both the local rule based and some of the identity based access control information may be generated by a local authority in the form of an auxiliary vector, which is bound to the identity certificate. The assemblage of information contained in the identity certificate and in the auxiliary vector will be referred to as the COMPOSITE ID. The "Local Domain Authority Identifier" field in the identity certificate identifies the source of the additional access control information to the SDNS user/component.

1.3.5 *Impact of Store-and-Forward Key Formulation*

The store-and-forward characteristic of electronic mail (E-Mail) introduces a number of special issues which do not apply to an environment in which peer entities communicate directly in real time. Originator and recipient user agents (UAs) do not communicate in real time. As a result, the information contained in a recipient's identity certificate and auxiliary vector (as posted on a server or bulletin board system) must be sufficient to allow an originator to perform any desired access control checks.

In the context of store-and-forward communication, it is not practical, in general, to carry out a distinct mutual association

validation step within PAA, subsequent to the certificate exchange. Each transmitted message is an autonomous entity, outside the context of any bilaterally-accepted association. Therefore, an originator must make any access control decisions without a prior validation step. Transmission takes place without prior, timely assurance of an intended peer's participation as the actual recipient of the transfer.

1.4 Four-Tiered Model for Access Control

The four-tiered model, described in subsequent sections of this paper, serves as a vehicle for defining the types of information necessary for making an access control decision. Depending on appropriate policy, with the exception of the first tier, not all tiers must be used by all SDNS components.

The four-tiered model consists of:

- **Partition Tier** - An SDNS access control division, at the highest level, of the population into discrete groups.
- **Partition Rule Based Access Control Tier** - Expression of the SDNS access control information which represents a policy common to all entities in the same partition.
- **Local Rule Based Access Control Tier** - Expression of the SDNS access control information which represents locally applicable policies or rules which govern the access to resources owned or administered by specific organizations.
- **Identity Based Access Control Tier** - Expression of the identity of the peer and additional information about the kinds of associations that are allowed with that peer.

2. PAA, PAE and THE FOUR-TIERED MODEL

2.1 Introduction

This section describes the processes of PAA and PAE and amplifies the four-tiered model. The relationship between PAA, PAE, and the four-tiered model is examined. Several figures are used to describe these processes, their operation, and their interoperation.

2.2 Relationship of PAA, PAE, and the Four-Tiered Model

SDNS provides two processes for access control decisions. The first process determines whether communications can be initiated, and the second process provides continuing access control of data. These processes are PAA and PAE, respectively. Figure 1 represents one possible

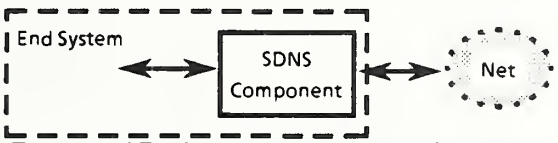


Figure 1. SDNS Relationship to the System

placement of an SDNS component relative to the network and the end system.[2]

Figure 2 illustrates the situation of the Key Management Protocol (KMP) calling PAA to perform its function; an exception to this is the store-and-forward-case. Two specific services in KMP make use of the PAA process. First, PAA is called upon to evaluate both peer's identity certificate information (Peer credentials received). All of the access control-relevant identity certificate information will be evaluated at this time. If no further access control information is to be exchanged, the PAA

2. This presentation relates only to the real-time case not to the staged delivery case. This is only one possible strategy, chosen for expository purposes, and it is not intended to constrain implementation alternatives.

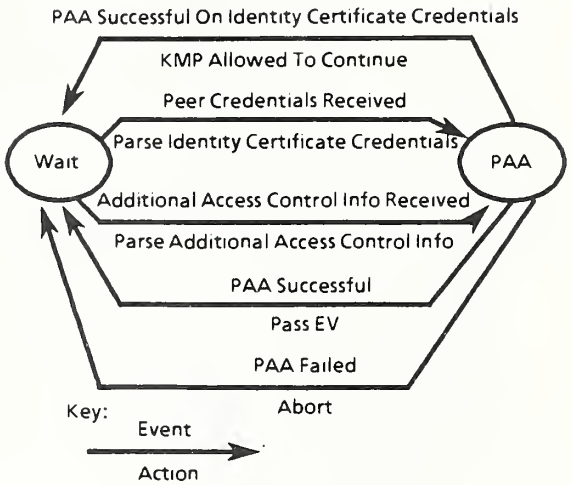


Figure 2. KMP/PAA State Diagram

process will yield a result (Process successful or Process failed) which will be returned to KMP. If additional access control information is to be exchanged, PAA will again be called upon to evaluate both peers' additional access control information (in the form of an Auxiliary Vector (AV)). (Additional access control information received.) Upon the completion of PAA a result will be returned to the KMP (Process successful or Process failed). It is this result which will determine the remainder of the processing which the KMP must perform.

Figure 3 illustrates the situation of the Security Protocol (SP) calling PAE to perform its processing. The SP acts as a gatekeeper for all data packets, both incoming and outgoing. When the SP receives a packet for processing it checks to see if the appropriate pointer is in place . If this pointer is not in place the SP will enter an error condition and drop the packet (some implementations may allow for an optional recovery attempt to be made; the packet being processed may be cached with the processing being transferred to KMP in an attempt to establish a secure association). At the appropriate time in the SP processing the PAE process will be called upon to perform its processing (PDU received for processing). The PAE process

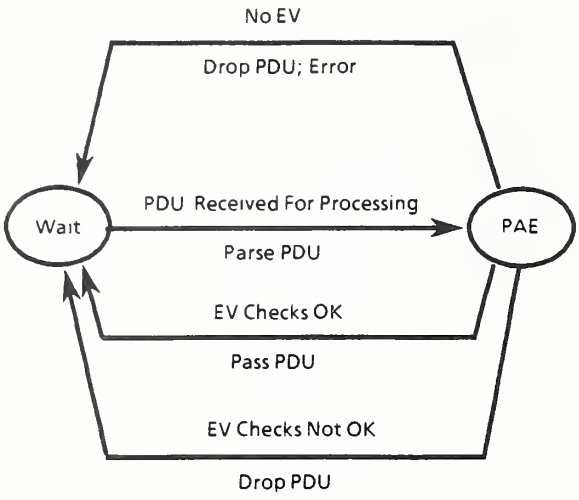


Figure 3. SP/PAE State Diagram

will yield a result (No EV, EV checks OK, EV checks not OK) which will be returned to the SP. It is this result which will determine the remainder of the processing which the SP must perform.

2.2.1 *Peer Access Approval*

Peer Access Approval is the process by which a sender/recipient uses a particular implementation of the four-tiered model to establish an EV. Figure 4, a top-level diagram of the four-tiered access control model's evaluation process, illustrates the evaluation process for both an initiator and recipient peer entity. The evaluation process depends on whether the peer entity is initiating or receiving the initial message, and also on whether the peer is functioning in the store-and-forward or real time mode. For an initiator, the PAA process begins when the local process discovers that an EV does not exist for an outbound PDU and concludes with the security service option negotiation. Correspondingly, the PAA process for a recipient begins when the local process receives an establishment request for an association and also concludes with the security service option negotiation.

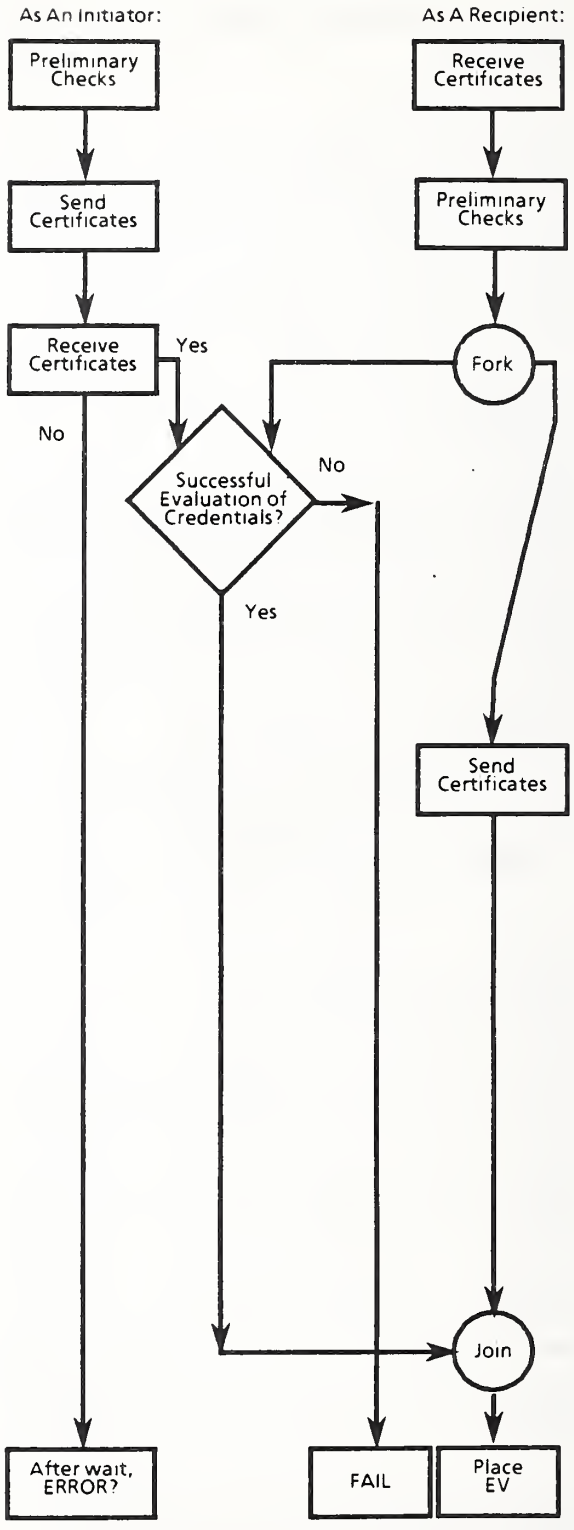


Figure 4. PAA Process

In the real time example, some preliminary checks can be made by the initiating side (e.g., is the peer request to the destination allowed ?). The identity certificate will then be sent to the destination SDNS component and the initiating SDNS component will await the arrival of its peer's identity certificate. If, after an appropriate wait, the destination SDNS component's certificate has not been received, a time-out will occur and an error message will be generated. If an identity certificate is received from the destination SDNS component then the PAA checks can occur.

The recipient process begins when the originator's identity certificate is received. The recipient can perform some preliminary checks such as, "Do I wish to communicate based on what I now know about this ID ?" and (depending on the local security policy) either before, during, or after the PAA checks, the identity certificate will be sent to the initiator.

The PAA checks begin with verifying that the received certificate's contents are valid. Following this, a check verifies that both certificates are in the same partition and then compares the remaining tiers in the four-tiered model (PRBAC, LRBAC, and IBAC tiers) as appropriate. If any of the PAA checks fail, the communication is aborted. In all other cases, the resulting PAA evaluation of the peer's tier values are placed (where appropriate) in the EV for enforcement during PAE. PAE will take the applicable PDU information (e.g., security label) and compare it with the information in the EV. The EVs used by each peer may be different based upon the security policy of the peer. A simple example is where one peer employs LRBAC while the other does not.

Figure 5 introduces the PAA process relative to the identity certificate exchange and generation of the EV. Upon receipt of the other peer SDNS component's identity certificate the evaluation by the four-tiered model will begin. If further identity information is required an auxiliary

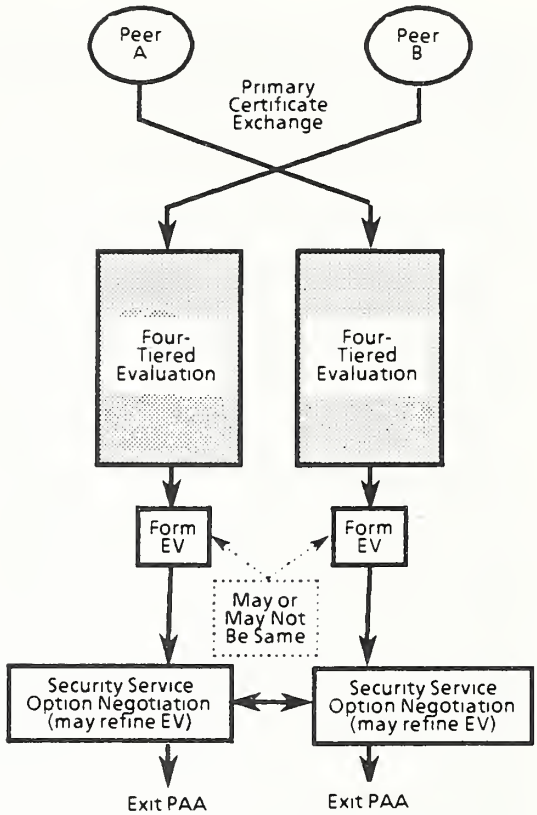


Figure 5. Evaluation of Four-Tiered Information

vector may be used (Section 1.3.4) or an external authority may be consulted. Upon the successful completion of the four-tiered model's evaluation (i.e., some form of communications at some classification level is permitted) an EV is formed. Depending on local security policies, the EVs at the two end points may or may not be identical. Once the EV has been formed a security service option negotiation takes place, possibly further restricting the EV. The EV is not used during the security service option negotiation exchange, this entire exchange is out of band with respect to the access control policy.).

There is no predetermined order for evaluating the lower three tiers of the model; tier one, however, must always be implemented. Each tier is independent and can be omitted if security policy dictates. The partition must be known in order to interpret all tier two, tier three, and tier

four data. If tiers two and three are interdependent then tier two must be evaluated if tier three is evaluated.

The PRBAC, LRBAC, and IBAC tier definitions permit a null value to be placed in the identity certificate whenever, for a specific tier, the local policy does not support that tier. As an alternative, it seems entirely reasonable that an entity could place information in any uninterpreted or unevaluated tier of its ID for interpretation as needed by the peers with which it communicates without implying that the accountable entity itself does any enforcement at the tier. In other words, the set of tiers which an accountable entity enforces may be independent of the tier information provided in its own ID. For example, all accountable entities' IDs should contain a representation of their identity, whether or not they are configured to perform IBAC enforcement, because the information might be needed by their peers for the peers' own IBAC information checking purposes and the information is definitely required for authentication purposes.

2.2.2 *Peer Access Enforcement*

The Peer Access Enforcement process provides continuous enforcement of the access control decision made in PAA. The enforcement mechanism becomes involved when the data is sent between peer entities. The Security Protocol (SP) is assigned the task of determining the appropriate time to call the PAE process, shown in Figure 6. The first thing PAE will attempt to do is find the appropriate EV for that specific PDU which it is operating on. If no EV can be found an error condition occurs and PAE is exited. If the appropriate EV is found then it is compared with the PDU's label. The result of the PAE process (either pass or fail) is then passed back to the SP as PAE is exited. The PAE determines whether or not the labeled PDU falls within the set of permissions represented by the EV. The PDU is either permitted to pass or not.

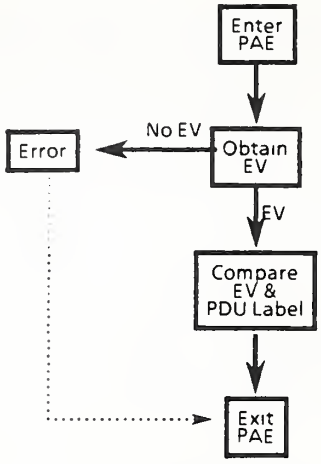


Figure 6. PAE Process

2.3 Individual Tier Descriptions

This section describes the four tiers in greater detail. The four-tiered model consists of:

- Partition
- Partition RBAC
- Local RBAC
- IBAC

"Partition RBAC" and "Local RBAC" were chosen to identify the authorities responsible for establishing the access control policy at that tier. Figure 7 shows a general decision process applicable for all layers of the ISO model (with the possible exception of Layer 1).

The four-tiered model is valid for ISO Layers 2, 3, 4, and 7, in addition to being appropriate in both classified and sensitive applications. An important point regarding the model is that it requires that tier one must be examined first. Subsequent tiers can be examined in any order, as appropriate for that specific SDNS component. Not all of the lower three tiers need to be considered in every SDNS application. It should be noted that if any interdependencies exist between tiers two and three, any subsequent evaluation of tier three requires that tier two also be evaluated. As in all cases where

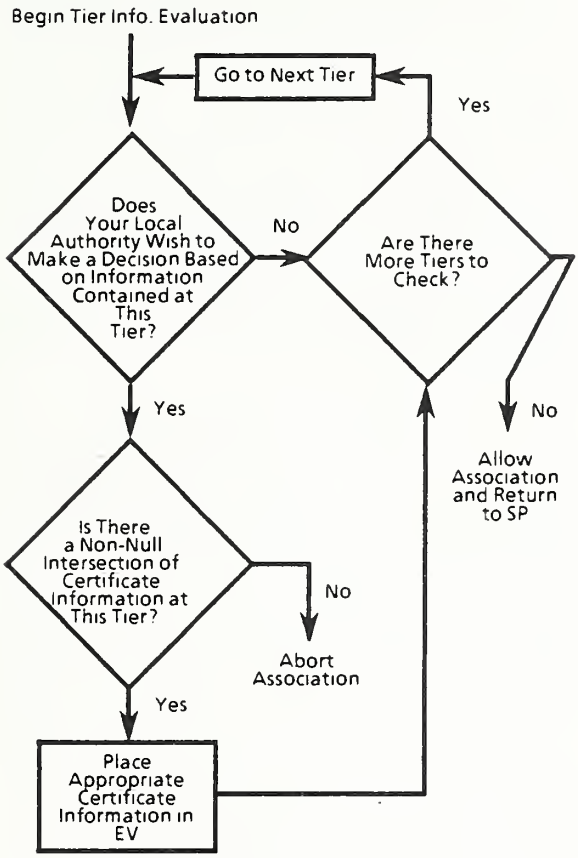


Figure 7. Generic Tier Process

a particular tier of the four-tiered model is implemented, the PAA checks must pass for that tier's values. If the PAA checks fail, then the association is disallowed.

2.3.1 *Partition*

While many different SDNS entities will want to intercommunicate, there are other groups of entities that do not desire and have no need to communicate. In fact, they must limit their communication to their group. The purpose of the partition level of the four-tiered model is to ensure that these groups can only communicate internal to their group. Once this separation into different partitions occurs, every SDNS entity enforces the separation and it cannot be circumvented.

The Partition tier separates entities into distinct groups. An accountable entity can be a member of only one partition per identity certificate. If the identity certificates do not share the same partition identifiers, then the PAA process does not allow communication. If an SDNS accountable entity needs to communicate with other partitions, a separate identity certificate is required for each partition. In special circumstances, when inter-partition communications are required, there will be a need for intermediaries holding identity certificates in multiple partitions and able to translate between those partitions' rules.

2.3.2 Partition Rule Based Access Control (PRBAC)

SDNS tier two access control is dependent upon the partition specified at tier one, and is uniquely defined for that given partition. The model recognizes, at the second tier, that the entities within a partition must have a common set of rules for the interpretation of the subsequent identification information. The rules are the expression of the access control policy. In order to enforce the common policy, there must also be a common representation of the attributes possessed by the entities in the partition.

Examples of PRBAC security policies are the mandatory security policies of the Department of Energy and the Department of Defense, both of which are lattice models. A DoD example of a possible tier two ID field structure is the following:

HIERARCHICAL:
- Security Level

NON-HIERARCHICAL:
- Compartments
- Markings

In this example, entities possess clearances to process data of certain classifications among the range of Top Secret, Secret, Confidential and Unclassified. The policy applied that an entity

cannot process data at a higher classification level than its clearance allows. This policy can be represented as a set of rules relating clearances and classifications.

Different interpretations of the lattice model will require separate partitions. Policies that do not fit the lattice model will be accommodated. Application of tier two for the sensitive, but unclassified market is left open for further study.

2.3.3 *Local Rule Based Access Control (LRBAC)*

Within a partition, there may be a group of entities that must satisfy security policies other than those expressed in the PRBAC tier. The LRBAC tier information permits the application of these "local" policies within a portion of a partition. This tier provides a finer definition of access characteristics that does not exist at the second tier. Local authorities establish the policy for the use and interpretation of local RBAC information. There may be several groups within the same partition that have local policies defined. These groups are included in the same partition because entities in the groups must communicate with other entities not in the same group. The entities share PRBAC policy (but not an LRBAC policy).

The use of the LRBAC tier occurs when an organization needs to enforce certain rules and interpretations beyond PRBAC, when communicating within that agency without preventing communications with entities outside the agency. This definition is not appropriate at tier two, but is meaningful at tier three. In this example, the specific organization is the local authority or administration responsible for the representation of the policy at this tier. When entities not sharing the same local authority communicate, it is generally assumed that the LRBAC tier information is meaningless. (There may be cases where policy translation between local authorities is meaningful, but this concept is outside the scope of this document.) For example, consider disjoint authorities which control the access information for their own domains. When entities wish to communicate between these domains they may do so as long as there is no information constrained by LRBAC (e.g., there is no compartmented data specific to that domain). Where there are equivalent compartments in different domains, an intermediate system which understands (has identity certificates for) both domains must be used, and it must translate compartment definitions based on pairwise agreements outside SDNS.

2.3.4 *Identity Based Access Control (IBAC)*

The remaining access control information is referred to as IBAC. Tier four, within SDNS access control, allows a local authority to specify identity based controls. An access control decision may be based on identity information in conjunction with the information from other tiers. However, an access control decision may be based solely on the identity of that entity. The granularity of IBAC, and of its supporting authentication functions, depends on the communication protocol layer at which an SDNS component operates. An example of information at this tier is an access control list. That is, a list that states with which other end systems are allowed to communicate. The primary certificate and auxiliary vector are the only sources of identification data for SDNS IBAC decisions.

There are at least two additional methods available whenever more identity information than that contained within the COMPOSITE ID is required. The first approach requires communicating with a third party in real-time by one or both of the parties in the communication. The second approach negotiates the additional IBAC information between the parties of the association.

The identity certificate used by a Layer 3/4 SDNS component represents an end system entity. If the end system's identity is represented in the certificate via a network address (physical

or logical, as used in a given network), or if the SDNS component can map between addresses and COMPOSITE ID representations with high assurance, an SDNS component directly connected to a single end system can verify consistency between emitted source addresses and the COMPOSITE ID representation. If the COMPOSITE ID representation of the identity information is not in the form of addresses, or if multiple distinct entities (e.g., "red networks") are located behind SDNS components, the level of consistency checking which an SDNS component can perform on emitted source addresses is limited or nonexistent.

2.4 Enforcement Vector

Within the SDNS access control framework the PAE process enforces the results determined by the PAA procedure. The PAA process results are represented by the Enforcement Vector, and held for the length of each association. The information within an EV can be separated into three categories. These categories are:

- **PDU identifier components**; identifies which existing EV is relevant for a specific association;
- **PDU filter components**; specifies the criteria against which a PDU is checked so that a pass/no pass decision may be made (i.e., results of PAA);
- **Miscellaneous components**: administrative information about the association.

The EV, at each peer, contains the following information for the association, depending upon the tiers used:

- Partition RBAC Information
- Local RBAC Information
- IBAC information

2.5 PDU Security Label Description

Each PDU presented for transmission may contain a security label. (Labels are a negotiable option and are not required in all cases.) Each PDU's security label is compared with the EV which allows the PAE process to enforce the established security policy on the PDU. For those end systems that do not provide a security label, the SDNS component will supply a predefined security label for PDUs sent from that end system. The PDU's security label does not have to match the entire EV exactly, but it must be a proper subset of the accountable entities' permissions.

3. OSI LAYER SPECIFIC ACCESS CONTROL FACTORS

Whereas the previous sections have presented the general access control structure for the SDNS, this section will examine some of the access control characteristics that are OSI layer specific. This is not an exhaustive presentation but illustrative of applying access control at the various layers of the OSI model supported by SDNS.

3.1 Layer 2

The data link layer transforms the raw transmission facilities provided by the physical layer into an apparent error free communications circuit to the layers above it. Layer 2 functions include the grouping and/or the adding of physical-layer bits into the Physical-layer Service Data Unit (PSDU) permitting the detection, correction, retransmission, and flow control functions of Layer 2 to be implemented. In addition, several different classes or qualities of service options, resulting in different performance and cost parameters, are available at the data link layer.

At Layer 2, the SDNS provides access control and authentication to the entities represented by the PSDU. A Layer 2 the SDNS component provides these services to data communications equipment using the identity definitions normally associated with the PSDU. The data link layer is highly dependent upon the media or actual physical-layer structure. Each unique physical layer structure requires different PSDU coding techniques and frame definitions. Therefore, there are various implementations of the SDNS layer 2 component for each of the media supported.

3.1.1 *Tiers Applicable to Layer 2*

- **Partition Tier (Tier 1)** - This tier is always defined.
- **PRBAC Tier (Tier 2)** - Defines the security level of the association.
- **LRBAC Tier (Tier 3)** - Some broadcast-type media Layer 2 implementations may use this tier.
- **IBAC Tier (Tier 4)** - The identity of a Layer 2 SDNS accountable entity is as a peer on a virtual link.

3.1.2 *Layer 2 PAE*

An implicit security label (resulting when a permitted connection implies a specific label) will generally be used on link applications because an a priori-determined security level will exist. In applications requiring an explicit label, the SDNS component will provide an appropriate label. The PAE process will basically monitor IBAC information at this layer. The EV is used to show that the link is legitimate and because of the single level assumption that all data may pass.

3.2 Layers 3/4

At the network-layer, end systems and intermediate systems are the peers between which SDNS access control services are applied. The accountable entities distinguished with regard to differing access rights are systems (either end system or intermediate system), not processes or individual users within those systems. At the transport-layer, the appropriate granularity for access control services depends on where within the transport layer security mechanisms are integrated. For security mechanisms integrated towards the bottom of the transport-layer, the appropriate access control granularity for such services is at the end system. Layer 4 may either apply access control on end system pairs or on transport connections. Subsequent subsections identify the tiers, labeling, and other attributes that are

used by the PAA and PAE processes at Layers 3/4.

3.2.1 *Tiers Applicable to Layers 3/4*

- **Partition Tier (Tier 1)** - This tier is always defined.
- **PRBAC Tier (Tier 2)** - Applicable in most classified implementations.
- **LRBAC Tier (Tier 3)** - Optional in all applications. Its use will most likely require the use of the auxiliary vector technique or other such alternatives to acquire all the necessary access control information.
- **IBAC Tier (Tier 4)** - As in tier 3 (LRBAC), extensions to the basic authentication process may be required to acquire all the necessary access control information.

At tier 3, LRBAC, the accountable entities' ID may also include the environmental, the accreditation, and the certification information of the end system.

3.2.2 *Layer 3/4 PAE*

Both implicit and explicit security labels will be available at Layers 3/4. PAE will enforce the security policies established for each Layer 3/4 SDNS component. These policies are realizable in the sense that they can, on a tier-wise basis, uniformly operate on the identity fields within the COMPOSITE ID in order to establish an EV for the association. Each security label will be evaluated against the established EV for the association.

3.3 **Layer 7 E-Mail**

The access control issues and mechanisms discussed in this section relate to electronic mail and are not necessarily applicable to the protection of other application-layer services (e.g., File Transfer Access Management (FTAM)). Subsequent consideration of other application-layer services may result in a definition of different access control services and mechanisms, consistent with those services' different characteristics. In particular, the store-and-forward delivery characteristic introduces a number of special issues which do not apply to an environment in which peer entities communicate directly in real time.

Users are "trusted" to process and correctly segregate information at any level up to and including their highest clearance. This does not necessarily imply that the same level of trust is appropriate for the end system on which a user processes data. The absence of interconnect rules between human users does not imply that no rule based mechanisms are appropriate. While it is legitimate for a TS-cleared user to send a message to a SECRET-cleared user, such a message should not contain any information designated with classification higher than SECRET. A sending SDNS UA can collect clearance information from certificates of a message's designated recipient, compute the intersection with the sender's privileges, and display that information to a sending user. (Note, however, that this function requires that recipients' certificates be cached or collected in real time, precluding implementations which let users request that transmission be performed as a "background" activity.) Further, the UA can verify the relation between the level provided in the certificate and the level at which the UA process is executing. For example, a single-level UA running at the TS level should not transmit mail to a recipient whose certificate indicates SECRET or lower clearance, although a multilevel UA spanning the TS and SECRET levels could transmit SECRET mail to such a recipient. SDNS E-Mail components will check the security label fields carried in messages against the clearances of the intended recipient as indicated in their certificates.

It is functionally desirable to keep end system interconnect rules and user IDs totally separate, allowing user mobility by disassociating a user's identity from any particular end system with which they may be associated. On the other

hand, preservation of end system risk indices may necessitate that the classification level(s) of information passed in E-Mail be constrained based on end system accreditation. It seems inappropriate to mandate that any such constraints be enforced by SDNS components operating independently below the application layer for two reasons: (1) this could unnecessarily constrain or preclude the use of SDNS E-Mail without an associated lower-layer SDNS; and (2) given store-and-forward delivery, no layers below E-Mail possess the end-to-end, communicating, peer-to-peer characteristics which are necessary for SDNS components to guarantee enforcement without relying on processing performed within non-SDNS intermediate entities (e.g., mail staging systems).

User-level COMPOSITE IDs will contain one of two separate classes of clearance/accreditation information: a first class (USER DATA) specifying the user's clearance information, and a second class (END SYSTEM DATA) reflecting those end system characteristics (e.g., authorized sensitivity levels and other inputs to risk index computations) which are needed in order to constrain dataflows appropriately. If the user moves to an alternate end system, the alternate end system's END SYSTEM DATA, rather than the home end system's END SYSTEM DATA, would become relevant. This implies that the process performed in order to activate a user's ID at an alternate end system must, in effect, associate USER DATA information with the applicable end system's END SYSTEM DATA.

3.3.1 *Tiers Applicable to Layer 7 E-Mail*

- **Partition Tier (Tier 1)** - This tier is always defined.
- **PRBAC Tier (Tier 2)** - Applicable in most classified implementations.
- **LRBAC Tier (Tier 3)** - Optional in all applications. Extensions to the basic authentication process may be required to acquire all the necessary access control information.
- **IBAC Tier (Tier 4)** - Required in all applications. Its use will most likely require the use of the auxiliary vector technique or other such alternatives to acquire all the necessary access control information.

3.3.2 *Layer 7 IDs*

Layer 7 IDs represent individual users (or their application agents). The first use of individual IDs will be for E-Mail. E-Mail IBAC determinations will be based on a process involving X.400 Originator/Recipient (O/R) Names and certificate ID fields identifying users. DoD Trusted Computer System Evaluation criteria (Orange Book) requirements for individual accountability (imposed at C2 levels and above) underscore the need for user identification at the granularity of named users or groups of named users.

As O/R Names (and identity certificate ID field components) are hierarchically qualified, it seems useful to provide analogous hierarchic qualification for IBAC features. For example, it could be appropriate to grant a particular user the right to send mail to anyone in organization A (without needing to exhaustively enumerate all members of organization A), as well as to user C (and user C alone) within organization B.

It is appropriate to consider the means by which peers are identified within SDNS (and specifically within CCITT's X.400 mail), as IBAC information fields in message headings must be associated with COMPOSITE ID fields. X.420 notes that peers may be identified in two ways: with an O/R name (a construct formally defined in X.411) or with a free-form name; for universal applicability in SDNS, use of the O/R Name is assumed. Three variant forms of O/R Name are defined, but the latter two are intended for special purposes (support for X.121 addressing or Telex interoperability), and the first variant is clearly the appropriate choice for consideration by SDNS. The first variant O/R Name form is defined as follows, where square brackets enclose those attributes which are optional:

O/R Name ::= Country Name
Administration Domain Name
[Private Domain Name]
[Personal Name]
[Organization Name]
[Organizational Unit Names]
[Domain-defined attributes]

The use of Administration Domain Name and where appropriate, Private Domain Name facilitates interoperability in advance of a directory service function spanning all domains. Administration domains are hierarchically qualified within countries, and private domains are hierarchically qualified within administration domains.

One or more of Private Domain Name, Personal Name, Organization Name, and Organizational Unit Names must be selected. Use of domain-defined attributes is optional. In assignment of O/R Names to its own peers, a domain is free to select a set of attributes which best serve its own purposes, potentially by using domain-defined attributes instead of standard attributes, although use of standard attributes is preferred for interoperability reasons. For universal interoperability among Layer 7 SDNS peers in advance of servers which can translate between different forms, it is appropriate to constrain the set of acceptable O/R Name component combinations.

The O/R Name subcomponents have CCITT-supplied syntactic definitions (in Recommendation X.411). For example, a Personal Name is defined as one or more printable strings, made up of a surname along with (optional) given name initials, and generation qualifier (e.g., "Jr.").

For SDNS purposes, it is proposed that a peer's ID as represented in a COMPOSITE ID contain the following O/R Name components: Country Name, Administration Domain Name, Organization Name, Organizational Unit Names (this component may be null if Organizational Unit Names are not used within the particular Organization), and Personal Name. Note that this identifies a peer in terms of organizational affiliation, not mailbox address, and hence does not preclude user mobility.

3.3.3 *Layer 7 E-Mail PAE*

Both explicit and implicit security labels will be available for Layer 7 E-Mail on an as-required basis. The security policies and characteristics have parallels with Layers 3/4 as discussed in Section 3.2.2 above.

4. DEFINITIONS

ACCESS CONTROL – The prevention of the unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner.(ISO 7498/2)

ACCOUNTABLE ENTITY – The logical element, unit, or user representation, within a system, which can be positively identified and authenticated using the access control attributes contained within a primary certificate and (when used) an Auxiliary Vector.(SDNS)

ASSOCIATION – A communications interaction between peer entities, which may occur over either connection-oriented or connectionless communication services.(SDNS)

AUDIT – An independent review and examination of system records and activities in order to test for adequacy of system controls, to ensure compliance with established policy and operational procedures, and to recommend any indicated changes in control, policy and procedures.(ISO 7498/2)

AUTHENTICATION – The process of positively validating a claimed identity. Each authentication process has a corresponding degree of certainty as to the accuracy of the identity established. Both the authenticated identity and the degree of certainty are used by an access control process to select the correct subset of rights and privileges associated with that entity.(SDNS)

AUTHORIZATION – The granting of rights or the subsequent possession of rights.(ISO 7498/2)

AUXILIARY VECTOR (AV) – A holder for additional ID information which does not normally fit within the identity certificate or is optionally excluded from the identity certificate. The Auxiliary Vector is bound to the identity certificate.(SDNS)

COMPOSITE ID – The assemblage of information contained in the identity certificate and the auxiliary vector.(SDNS)

DATA ORIGIN AUTHENTICATION – The corroboration that the source of the data received is as claimed.(ISO 7498/2)

END SYSTEM – Any computer-based system connected to the network and containing the necessary protocol interpreter software to initiate network access and carry out information exchange across the communications network.(Trusted Network Interpretation)

ENFORCEMENT VECTOR *(EV)* – The resulting information from a successfully completed PAA process which denotes the evaluation of the two communicating entities' security policies. This vector is an inter-component binding of security attributes of the requested communications association. Creation of an Enforcement Vector implies that peer communications is permissable under the limitations that are represented in the Enforcement Vector.(SDNS)

ENTITY – The logical element, unit, or user representation, within a system, on behalf of which an SDNS component provides security services.(SDNS)

EXPLICIT SECURITY LABEL – A precisely defined set of security attributes, explicitly and incorruptibly appended or prepended to a PDU, which qualify its sensitivity, specify handling and distribution caveats, and end-system specific requirements. The PDU explicit security label is used as an aid in carrying out a security policy. (SDNS)

FOUR TIERED MODEL – A logical representation describing an entity's security policies and identity. Both rule and identity based access control policies are capable of being described.(SDNS)

IDENTITY-BASED SECURITY POLICY – A security policy based on the identities and/or attributes of users, a group of users, or entities acting on the behalf of the users and the resources/objects being accessed.(ISO 7498/2)

IDENTITY CERTIFICATE – It is the data element that contains the identification data, along with other information, which is exchanged with an originator/recipient pair to provide both access control and authentication information.(SDNS)

IDENTITY DATA – The portion of the identity certificate which describes the end entity in the context of the four tiered model. The strawman definition of this information includes such things as NAME, DAO code, SECURITY LEVEL, and other tier relevant information.(SDNS)

IMPLICIT SECURITY LABEL – A precisely defined set of security attributes for a PDU which qualify its sensitivity, specific handling and distribution caveats, and end-system requirements. The set of security attributes are held by each security service entity processing the PDUs and these security attributes are referenced through an association with the PDU. The PDU's implicit security label is used as an aid in carrying out a security policy.(SDNS)

INTERMEDIATE SYSTEMS – System which converts packets from one protocol to another.(Computer Networks)

INTERNAL HOST – An addressable entity which resides in the SDNS component and is used for such things as SDNS system management functions.(SDNS)

LOCAL DOMAIN AUTHORITY IDENTIFIER – The identity of the controlling authority for a local SDNS administrative community, responsible for access control information. (SDNS)

PAA – Peer Access Approval. The process which yields an Enforcement Vector by interpreting the ID information of the two communicating peer end points.(SDNS)

PARTITION IDENTIFIER – A single entity (possibly a number) which at the highest level divides the entire user population into discrete segments each sharing a common high level security policy.(SDNS)

PAE – Peer Access Enforcement. The application of the Enforcement Vector on a per PDU basis, which determines whether or not a particular PDU can be sent or received by an end system.(SDNS)

PEER ENTITY AUTHENTICATION – The corroboration that a peer entity in an association is the one claimed.(ISO 7498/2)

RULE BASED SECURITY POLICY – A security policy based on a set of rules interpreted in a common fashion across a set of interoperating SDNS components. These rules usually rely on a comparison of the sensitivity levels of the resources being accessed and the possession of corresponding attributes of accountable entities acting on behalf of users.(SDNS)

SDNS COMPONENT – A name which identifies the abstract representation of any SDNS security product capable of providing one or more SDNS security services.

5.0 ABBREVIATIONS

AV	Auxiliary Vector
EV	Enforcement Vector
FTAM	File Transfer Access Management
IBAC	Identity Based Access Control
LRBAC	Local Rule Based Access Control
MLS	Multilevel Secure
NAK	Negative Acknowledgment
PAA	Peer Access Approval
PAE	Peer Access Enforcement
PDU	Protocol Data Unit
PSDU	Physical Layer Service Data Unit
PRBAC	Partition Rule Based Access Control
RBAC	Rule Based Access Control
SDNS	Secure Data Network System
UA	User Agent

ACCESS CONTROL SPECIFICATION

Rev 1.0

25 July 1989

Table of Contents

Associated Addendum:

Addendum 1 - Access Control Information Specification (ACIS); SDN.802/1

SECURE DATA NETWORK SYSTEM

ACCESS CONTROL SPECIFICATION 25 July 89

0 INTRODUCTION

0.1 About this specification

This purpose of this document is to provide a common basis from which devices implementing the access control service will be able to achieve interoperability. This document also identifies points of reference for users implementing the Secure Data Network System (SDNS) Security Protocols (SPs).

This specification must be applied in conjunction with specifications for other aspects of SDNS. As a basis for understanding this document, it is assumed that the reader is familiar with the SDNS Access Control Concept Document (SDN.801), revision 1.3.

0.2 Related SDNS specifications

SDN.301	SDNS Security Protocol 3 (SP3)
SDN.401	SDNS Security Protocol 4 (SP4)
SDN.601	SDNS Key Management Profile
SDN.701	SDNS Message Security Protocol
SDN.702	SDNS Directory Specification for Utilization with the SDNS Message Security Protocol
SDN.801	SDNS Access Control Concept Document
SDN.802/1	SDNS Access Control Specification, Addendum 1: Access Control Information Specification (ACIS)
SDN.902	SDNS Key Management Protocol; Definition of Services provided by the Key Management Application Service Element
SDN.903	SDNS Key Management Protocol; Specification of the protocol for services provided by the Key Management Application Service Element
SDN.906	SDNS Key Management Protocol; Attribute Negotiation

1. SCOPE and FIELD of APPLICATION

This specification gives a functional description of the SDNS access control system. It establishes a point of reference from which security protocols can make use of the access control service.

This specification also provides an overview of the Access Control Information Specification (ACIS). ACIS provides a uniform method for encoding access control information which is independent of any particular security policy. It also provides a standard algorithm for interpreting and comparing access control attributes.

2. ABBREVIATIONS

ACIS	Access Control Information Specification
ACWG	Access Control Working Group
AV	Auxiliary Vector
CRL	Certificate Revocation List
EV	Enforcement Vector
IBAC	Identity Based Access Control
ISO	International Standards Organization
KMP	Key Management Protocol
KMAP	Key Management Application Process
LAW	Local Authority Workstation
LDA	Local Domain Authority
LRBAC	Local Rule Based Access Control
MEK	Message Encryption Key
MIB	Management Information Base
MSP	Message Security Protocol
NSAP	Network Service Access Point
OSI	Open Systems Interconnect
PAA	Peer Access Approval
PAE	Peer Access Enforcement
PCID	Primary Certificate Identifier
PDU	Protocol Data Unit
PRBAC	Partition Rule Based Access Control
RBAC	Rule Based Access Control
SDNS	Secure Data Network System
SP	Security Protocol
TEK	Traffic Encryption Key
UA	User Agent

3. ACCESS CONTROL INFORMATION REPRESENTATION

The four-tiered model for access control, described in detail in the SDNS Access Control Concept Document (SDN.801), serves as a vehicle for defining the types of information necessary for making an access control decision. At a high level the four-tiered model consists of:

- **Partition Tier** - An SDNS access control division, at the highest level, of the population into discrete groups.
- **Partition Rule Based Access Control (PRBAC) Tier** - Expression of the SDNS access control information which represents a policy common to all entities in the same partition.
- **Local Rule Based Access Control (LRBAC) Tier** - Expression of the SDNS access control information which represents locally applicable policies or rules which govern the access to resources owned or administered by specific organizations.
- **Identity Based Access Control (IBAC) Tier** - Expression of the identity of the peer and additional information about the kinds of associations that are allowed with that peer.

Primary certificates are interpretable by all SDNS components sharing a common universal. Primary certificates are anticipated to carry access control information related to the Partition Tier, to the PRBAC tier, and to the IBAC tier. Primary certificate formats are defined uniformly.

AVs have local significance within the set of SDNS components under the jurisdiction of a common local domain authority, and are used to reflect attributes of the communicating accountable entities with which the AVs are associated. It is anticipated that auxiliary vectors will be used to carry access control information related to the LRBAC and IBAC tiers. LRBAC and IBAC information carried in AVs has the property that its syntactic and semantic significance is determined by the local domain authority.

3.1 Four-Tiered Model Information

In this subsection, we examine the relationship between the tiers of the four-tiered model and the points where information relevant to the decisions made in each of the model's tiers is represented. The next subsection will consider the detailed representation of the data relevant to access control as it occurs in primary certificates, in auxiliary vectors, and in PDUs. It should be recognized that each of these constructs also contains additional data (e.g., KMS-specific information and certain labeling information interpreted only by end systems) which is not within the scope of SDNS access control.

3.1.1 *PARTITION TIER*

Information relevant to the Partition tier is represented in the primary certificate. The test for common partition membership between peers attempting communication is performed wholly by PAA based on comparison between certificate contents; no PDU fields are relevant to this tier.

Information relevant to the partition tier includes:

partition code

3.1.2 *PRBAC TIER*

Information relevant to the PRBAC tier is represented in the primary certificate and may occur in the headers of the PDUs processed by SDNS components. Primary certificate space considerations may make it necessary to carry a proper subset of PRBAC information in AVs, but this course of action would limit interoperability across jurisdiction boundaries.

Information relevant to the PRBAC tier includes:

- Hierarchical information (e.g.,security level(s))
- Non Hierarchical information (e.g.,channel,control system)

3.1.3 *LRBAC TIER*

Information relevant to the LRBAC tier is represented in the AV (those communities which do not make use of an AV and require LRBAC will have to develop another method for representing and communicating the LRBAC information to their corresponding peers) and may occur in the headers of the PDUs processed by SDNS components. (Exception: the local domain authority ID corresponding to a particular AV is carried in the associated primary certificate's Local Domain Authority Identifier field: since an associated primary certificate is available for interpretation whenever an AV is processed, this does not constitute a limitation.) The ACIS encoding mechanism is highly suggested, but not mandated, as a representation for LRBAC information and its interrelationships with PRBAC information.

In some instantiations, LRBAC is an extension of the PRBAC tier, for this case some of the PRBAC information may need to be repeated in the AV to accurately represent this extension. When this is true, consistency must be maintained between the PRBAC information contained in both the primary certificate and the AV. Any PRBAC information placed in an Enforcement Vector (EV) should accurately reflect the PRBAC information which appears in the primary certificate, any of the PRBAC information which is repeated in the AV must be consistent with the PRBAC information contained in

the primary certificate. Nothing can be added to the EV (in the way of contradictory PRBAC information) due to the contents of an AV, which was not allowed for by the primary certificate.

Information relevant to the LRBAC tier includes:

Representation of entity LRBAC attributes, including (for example):

- compartments

3.1.4 *IBAC TIER*

Information relevant to the IBAC tier is represented in both the primary certificate and the AV. IBAC-relevant information may be represented in the headers of the PDUs processed by SDNS components. A standard set of IBAC is represented in the primary certificate.

Information relevant to the IBAC tier includes:

ASCII ID (as Originator/Recipient (O/R) name for layer 7 entities)

3.2 Access Control Portions of the Primary Certificate

This subsection examines the portions of the primary certificate which contain information that can be used in making an access control decision. These fields are to be contained within the primary certificate, being supplied by a central authority. Additional access control information may be placed within the locally generated AV, if it cannot be placed within the primary certificate and local doctrine allows this. Information is placed in the primary certificate to support two of the access control system's primary axioms: the first axiom states that a maximum amount of access control information must be contained within the primary certificate, this maximizes interoperability between different communities within a partition; the second axiom states that sufficient access control information must be contained within the primary certificate to allow for a meaningful access control decision to be made, this allows communication between communities having different local authorities or those communities which do not support the use of AVs.

The access control portions of the primary certificate must accurately and completely convey the attributes of the accountable entity which is being represented. At a minimum, the access control portions of the primary certificate must contain enough information to support the accountable entity's local policy constraints.

3.3 The Auxiliary Vector

3.3.1 *PURPOSE*

The purpose of the AV is to supplement the access control system information contained in the primary certificate. The AV presents a mechanism to locally define, format, and process access control data which is (a) highly volatile, (b) of use, interest, or constrained only to a small community, or (c) extensible beyond the available field lengths in the primary certificate.

3.3.2 *COMPOSITION*

The contents of the AV are variable, depending upon local network security requirements and policies for access control. The projected contents will include information binding the AV to the SDNS component's primary certificate, information which describes the identity, access permissions, and/or governing local policies assigned to that entity, and data integrity check fields which will ensure any attempt at AV data manipulation is detectable.

3.3.3 *AV Integrity*

As the AV carries access control information which is presumptively more detailed (a finer level of granularity) than that which is contained in the primary certificate. The level of system integrity provided the AV must meet the minimum level required by local network security policy for access control. The information contained in the AV must be protected in accordance with whatever policy governs the control of that information.

3.3.4 *APPLICATION TO PAA*

The Auxiliary Vector, as stated in paragraph 3.3.1, provides supplementary information to complete a highly granular PAA process. The Primary Certificate contains information used for access control decisions at the Tier 1 (Partition) and Tier 2 (PRBAC) PAA decision levels. The AV may contain some PRBAC Data (e.g.:, if this information is needed to accurately represent the tier 3 data which is an extension of the tier 2 data), Tier 3 (LRBAC) and Tier 4 (IBAC) level data. During PAA, the primary certificate is passed as part of the initial KMP exchange. After a Traffic Encryption Key is formed, and if the PAA partial results are positive, KMP will exchange the AV. Local policy will determine these requirements. Once the AV data has been fully passed, PAA will complete. If positive (PAA is passed), the formed EV will be applied during PAE. If PAA fails, the association is dropped.

4. ACCESS CONTROL FUNCTION DEFINITIONS AND SPECIFICATION

This section describes the access control functions, sub-functions, requirements, and interfaces, for the application of SDNS access control to both the direct real-time and staged delivery communications services. The processes of Peer Access Approval (PAA) and Peer Access Enforcement (PAE) are examined in detail. The relationships between PAA, PAE, KMP, and the SPs are explained. Only brief, functional descriptions of the SPs and KMP are given here, the reader is directed to reference the particular protocol specification desired for further details. Several figures are used to aid in representing these relationships, and along with the accompanying functional descriptions give a good layout for how access control fits into the overall SDNS. The modular allocation of functions described in this section is for expository purposes only and is not meant to imply any restrictions on the organization of actual implementations.

4.1 Peer Access Approval

The PAA process is the process which evaluates the peer security attribute information of the two communicating peer endpoints in an attempt to determine a commonly held set of security attributes . This commonly held set of security attribute information can be further pared down through the application of local configuration information to yield an Enforcement Vector (EV). It is this EV which the PAE process will use later to screen against PDUs.

The PAA process is called from the Key Management Protocol's (KMP) Key Management Application Process (KMAP)(the KMAP is defined in detail in the KMP documentation: SDN.902 and SDN.903) to perform its functionality, an exception to this is the Message Security Protocol (MSP) (explained in section 4.3). Two specific services of the KMAP make use of the PAA process; Exchange Credentials and Attribute Negotiations. Figure 4.1 shows this relationship.

The Exchange Credentials service will call PAA to evaluate both peers' primary certificate information. All of the access control relevant primary certificate information will be evaluated at this time. If no further access control information is to be exchanged, the PAA process will yield a result (PAA_Result) which will be returned to the KMAP. If additional access control information is to be exchanged the Attribute Negotiations service will call PAA to evaluate both peers' additional access control information (in the form of an AV). Upon the completion of PAA a result will be returned to the KMAP (PAA_Result). It is this result which will determine the remainder of the processing which the KMAP must perform. When any of the PAA checks fail, PAA_Result = Fail and any audit messages required by local policy will be generated as control is returned to the KMAP. The following

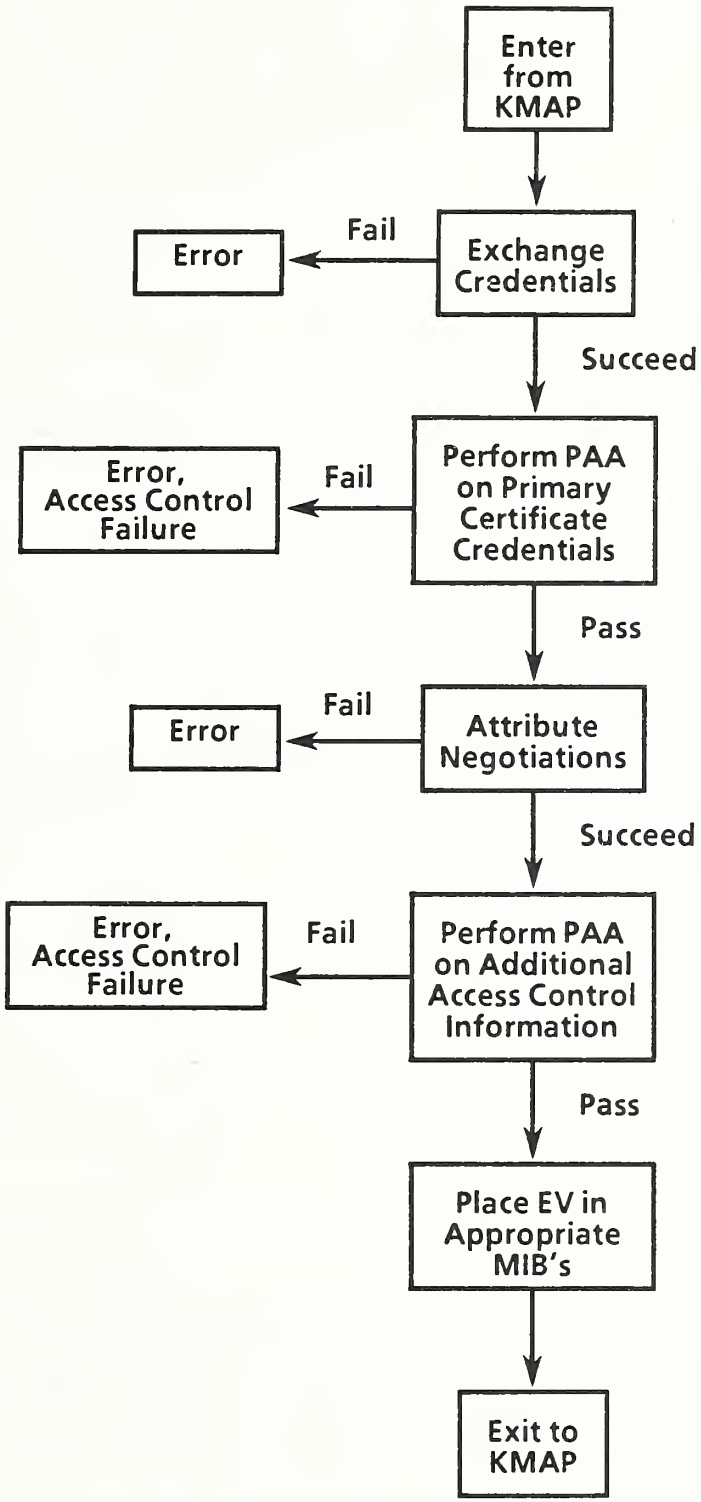


Figure 4.1. KMP/PAA Interrelationship

sections describe the details of the PAA process, figure 4.2 illustrates the functions which are described below.

4.1.1 *ENTER PAA*

Inputs: Local Primary Certificate
Remote Primary Certificate
Local Configuration Information
Local Auxiliary Vector (AV) (if using one)
CRL Version Number
Proposed Security Service Options (if initiator)

Outputs: Request to Check Universal

Functionality:

The Enter PAA function is the point at which the KMAP will transfer control over to the PAA process so that a comparison between peer security attributes can be made in an attempt to determine an allowable set of these attributes, which will be bound to the association being formed between accountable peer entities. The allowable intersecting set of security attributes will be represented in an EV, which will be used later in the PAE process. The Enter PAA function will call the Check Universal function to perform its processing.

4.1.2 *CHECK UNIVERSAL*

Inputs: Local Universal ID
Remote Universal ID

Outputs: Univ_Check = Pass/Fail

Functionality:

The Check Universal process ensures that the universal being used in both the local and remote primary certificates is identical. If the universals are identical then Univ_Check = Pass and the Decrypt Certificate function is called to perform its processing. If the universals are not identical then Univ_Check = Fail. When Univ_Check = Fail an audit message may be generated and the control of the processing will be returned to the KMAP.

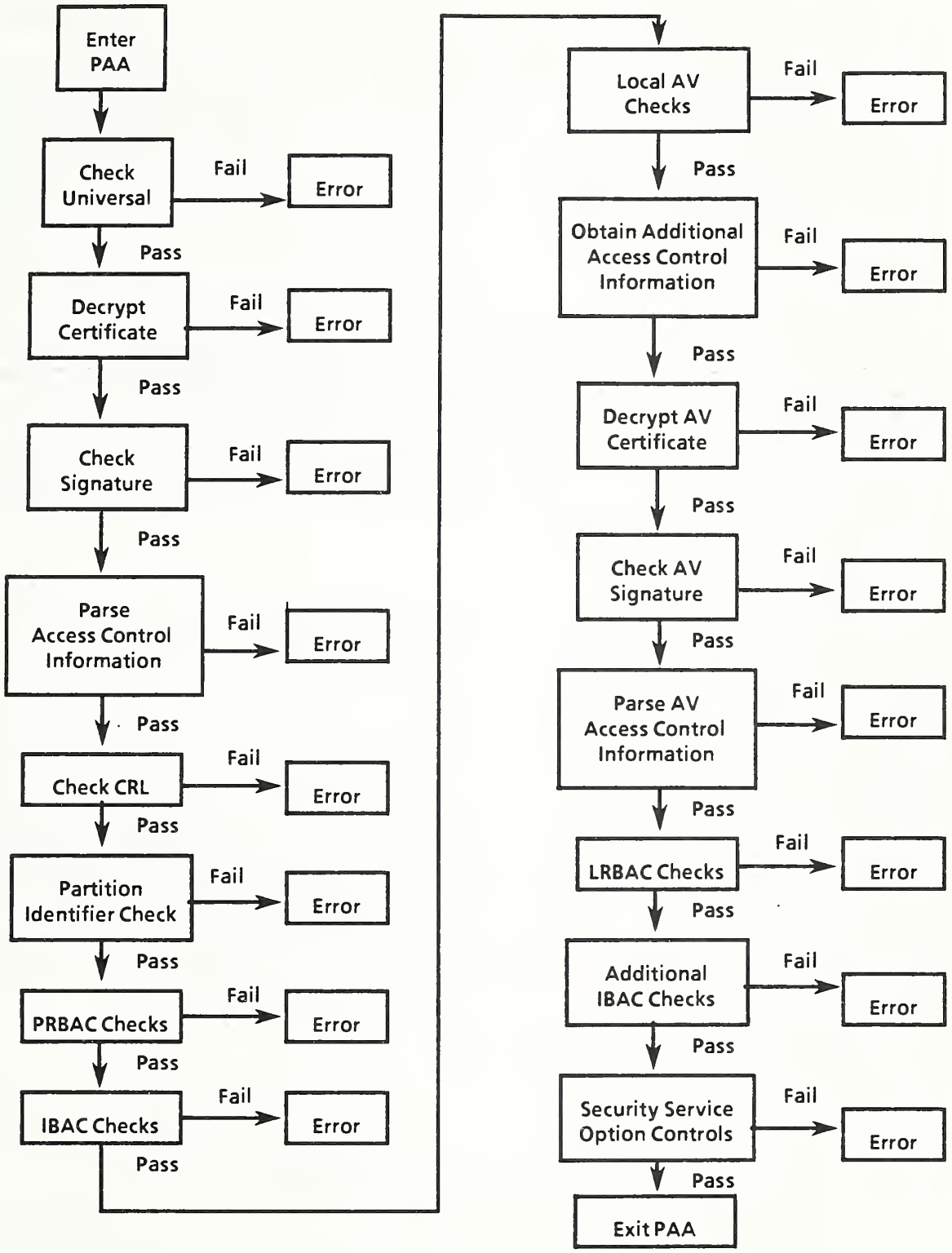


Figure 4.2. Peer Access Approval (PAA) Process

4.1.3 *DECRYPT CERTIFICATE*

Inputs: Encrypted Remote Primary Certificate
Universal

Outputs: Remote Primary Certificate
Decrypt_Cert = Pass/Fail

Functionality:

The Decrypt Certificate function will use the Universal to decrypt the remote peer's primary certificate. If the certificate is successfully decrypted then Decrypt_Cert = Pass and the decrypted remote peer's primary certificate is passed to the Check Signature function.(The decrypted Remote Primary Certificate will hereafter be referred to as the Remote Primary Certificate.) If the certificate cannot be decrypted then Decrypt_Cert = Fail, an audit message may be generated and the control of the processing will be returned to the KMAP.

4.1.4 *CHECK SIGNATURE*

Inputs: Remote Primary Certificate

Outputs: Signature = Pass/Fail

Functionality:

The Check Signature function will perform an integrity check on the remote primary certificate to insure that the contents haven't been tampered with. If the integrity check passes then Signature = Pass and the Parse Access Control Information function is called to perform its processing. If the integrity check fails then Signature = Fail, an audit message may be generated and the control of the processing is returned to the KMAP.

4.1.5 *PARSE ACCESS CONTROL INFORMATION*

Inputs: Remote Primary Certificate

Outputs: Parsed Remote Certificate's Access Control Information
Parse_Cert = Pass/Fail

Functionality:

The Parse Access Control Information function will parse the remote primary certificate and locate all the access control information to be used in later checks. If the required access control information is successfully parsed then Parse__Cert = Pass and the Check Certificate Revocation List function is called to perform its processing. If the parsing operation is unsuccessful then Parse__Cert = Fail, an audit message may be generated and the control of the processing will be returned to the KMAP.

4.1.6 *CHECK CERTIFICATE REVOCATION LIST*

Inputs: Local Configuration Information(or whatever medium the local implementation may use to store the CRL)
Remote Primary Certificate Identifier (PCID)

Outputs: CRL__Check = Pass/Fail

Functionality:

The purpose of the Check Certificate Revocation List (CRL) function is to make sure that the primary certificate used by the remote peer hasn't been reported as no longer valid. The Identifier of the remote primary certificate (PCID) is compared to the CRL of the local device, which is contained in the Local Configuration Information. If the PCID is not found on the CRL, then CRL__Check = Pass and the Partition Check function is called to perform its processing. If the PCID is found to be contained on the CRL, then CRL__Check = Fail, an audit message may be generated and the control of the processing will be returned to the KMAP.

4.1.7 *PARTITION IDENTIFIER CHECK*

Inputs: Local Primary Certificate Partition Number
Remote Primary Certificate Partition Number
KID

Outputs: Partition__Check = Pass/Fail
Entry in EV

Functionality:

The Partition Check is used to enforce the separation of SDNS accountable entities into distinct groups. Partition numbers will be uniquely assigned across the entire SDNS by the KMS. If the

partition numbers of both primary certificates are not the same then the Partition Check fails (Partition_Check = Fail), an audit message may be generated and the control of the processing will be returned to the KMAP. If the partition numbers match then the Partition Check passes (Partition_Check = Pass), and the PRBAC Checks function is called to perform its processing. It is at this point that the formation of the EV will begin, the KID assigned to the Init_TEK by the KMAP will be included in the EV to bind that EV and TEK together. The KID is placed in the EV at this point because this check is the only one which all SDNS components must perform when doing PAA.

When subscriber components are attempting to communicate with the KMS, the Partition Identifier Check is the last PAA check which they need to perform. If the received remote primary certificate indicates that it represents the KMS, through an explicit partition identifier assigned to the KMS, then PAA_Result = Pass and the control of the processing will be returned to the KMAP.

4.1.8 *PARTITION RULE BASED ACCESS CONTROL CHECKS*

Inputs: Local Primary Certificate PRBAC Information
Remote Primary Certificate PRBAC Information
Local Configuration Information

Outputs: PRBAC_Enforced = True/False
PRBAC_PAA = Pass/Fail
Entry in EV

Functionality:

The Partition Rule Based Access Control (PRBAC) Checks function performs two tasks: it determines if the PRBAC tier is being enforced by the local SDNS component; and then, if the PRBAC tier is being enforced , it performs the PRBAC PAA checks. These tasks are defined in the following two sections (4.1.8.1 & 4.1.8.2).

4.1.8.1 *PRBAC Enforced Check*

Inputs: Local Configuration Information

Outputs: PRBAC_Enforced = True/False
Entry in EV

Functionality:

The PRBAC Enforced Check determines if the PRBAC tier is being enforced by the local SDNS component. This function will examine the Local Configuration Information to see if the PRBAC tier is to be enforced. If the Local Configuration Information indicates that the PRBAC tier is to be enforced, then PRBAC_Enforced = True and the PRBAC PAA Check function is called to perform its processing. If the Local Configuration Information indicates that the PRBAC tier is not being enforced, then PRBAC_Enforced = False and the IBAC Checks function is called to perform its processing. When the PRBAC_Enforced = False the EV is set to indicate that the PRBAC tier is not being enforced.

4.1.8.2 *PRBAC PAA Check*

Inputs: PRBAC_Enforced
Local Primary Certificate PRBAC Information
Remote Primary Certificate PRBAC Information

Outputs: PRBAC_PAA = Pass/Fail
Entry in EV

Functionality:

The PRBAC PAA Check is responsible for evaluating the Local and Remote Primary Certificate's PRBAC Information. (Note: For this function to be performed the PRBAC_Enforced boolean must be in the "True" state.) This evaluation will determine if there is any PRBAC information (e.g., attributes, permissions) which is shared by both peers. If there are any PRBAC attributes which both peers hold in common, then PRBAC_PAA = Pass. PRBAC attributes which will be represented in a PDU's attributes, checked against in PAE, and are held in common between the peers are placed in the EV. Once the applicable PRBAC information is placed in the EV the IBAC Checks function will be called to perform its processing. If there are no PRBAC attributes which are held in common by both peers, then PRBAC_PAA = Fail, an audit message may be generated and the control of the processing will be returned to the KMAP.

4.1.9 *IDENTITY BASED ACCESS CONTROL CHECKS*

Inputs: Remote Primary Certificate IBAC Information
Local Configuration Information

Outputs: IBAC_Enforced = True/False
IBAC_PAA = Pass/Fail
Entry in EV

Functionality:

The Identity Based Access Control (IBAC) Checks function performs two tasks: it determines if the IBAC tier is being enforced by the local SDNS component; and then, if the IBAC tier is being enforced, it performs the IBAC PAA checks. These tasks are defined in the following two sections (4.1.9.1 & 4.1.9.2).

4.1.9.1 *IBAC Enforced Check*

Inputs: Local Configuration Information

Outputs: IBAC_Enforced = True/False
Entry in EV

Functionality:

The IBAC Enforced Check determines if the IBAC tier is being enforced by the local SDNS component. This function will examine the Local Configuration Information to see if the IBAC tier is to be enforced. If the Local Configuration Information indicates that the IBAC tier is to be enforced, then IBAC_Enforced = True and the IBAC PAA Check function is called to perform its processing. If the Local Configuration Information indicates that the IBAC tier is not being enforced, then IBAC_Enforced = False and the Local Auxiliary Vector Checks function is called to perform its processing. When the IBAC_Enforced = False the EV is set to indicate that the IBAC tier is not being enforced.

4.1.9.2 *IBAC PAA Check*

Inputs: IBAC_Enforced
Remote Primary Certificate IBAC Information
Local Configuration Information

Outputs: IBAC_PAA = Pass/Fail
Entry in EV

Functionality:

The IBAC PAA Check is responsible for evaluating the Remote Primary Certificate's IBAC Information. (Note: For this function to be performed the IBAC_Enforced boolean must be in the "True" state.) The evaluation will determine if the Remote IBAC Information passes the IBAC checks which are required by the policy of the local SDNS accountable entity. As an example, a host name may be checked against a locally maintained access control list. This information would be contained in a portion of the Remote Primary Certificate's IBAC Information and could be checked before the association would be allowed. The exact nature of these IBAC checks will be contained in the local SDNS component's Local Configuration Information.

If the IBAC checks are successfully completed (IBAC_PAA = Pass), any appropriate IBAC information needed for use in PAE will be placed in the EV. Once the appropriate IBAC information is placed in the EV the Local Auxiliary Vector Checks function will be called to perform its processing. If any of the IBAC checks are unsuccessful, then IBAC_PAA = Fail, an audit message may be generated and the control of the processing will be returned to the KMAP.

4.1.10 *LOCAL AUXILIARY VECTOR CHECKS*

Inputs: Local Primary Certificate LDA ID
Remote Primary Certificate LDA ID
Local Configuration Information
Auxiliary Vector MIB

Outputs: Common_LDA = True/False
Use_AV = True/False
Local Auxiliary Vector

Functionality:

The purpose of the Local AV Checks task is twofold; first it determines if an AV can (and will) be used for this association, and second it determines the correct AV to be used for this association. This function is broken up into three separate sections: the Local Configuration Information Check (4.1.10.1), the Local Domain Authority ID Check (4.1.10.2), and the Select Correct AV (4.1.10.3).

4.1.10.1 *Local Domain Authority ID Check*

Inputs: Local Primary Certificate LDA ID
Remote Primary Certificate LDA ID

Outputs: Common_LDA = Pass/Fail

Functionality:

The function of the LDA ID Check is to determine if the LDA identified in both the Local and Remote Primary Certificate is the same. The Local Domain Authority Identifier field within the Primary Certificate will contain the ID of the LDA associated with that Primary Certificate (assuming there is an LDA associated with this certificate, otherwise the field will be null).

If either the LDA ID found in both Primary Certificates is null or the LDA ID in both Primary Certificates is found to be different, then an AV is not used for this association, LDA_Check = Fail, and the Additional Access Control Information function is called to perform its processing. If the LDA ID is found to be the same in both Primary Certificates, then LDA_Check = Pass, and the AV Implemented Check function is called to perform its processing.

4.1.10.2 *AV Implemented Check*

Inputs: Local Primary Certificate AV_Required Flag
Remote Primary Certificate AV_Required Flag
Local Configuration Information

Outputs: Use_AV = True/False

Functionality:

The Auxiliary Vector (AV) Implemented Check determines if an AV will be used for the association which is being created. The AV_Required flag contained within the Primary Certificate will be used to identify whether or not an AV must always be used when communicating with a peer which shares the same LDA ID.

If the AV_Required flag is set in either the Local or Remote Primary Certificate an AV must be used for the association which is being formed, USE_AV = True, and the Select Correct AV function is called to perform its processing. If the AV_Required flag is not set in either of the Primary Certificates then the local SDNS component must use Local Configuration Information to determine

whether or not to return an AV. The determination mechanism which uses the Local Configuration Information should be consistent across an LDA. If the Local Configuration Information indicates that an AV should be used, then USE_AV = True, and the Select Correct AV function is called to perform its processing. If the Local Configuration Information indicates that an AV should not be used , then USE_AV = False, and the Additional Access Control Information function is called to perform its processing.

4.1.10.3 *Select Correct AV*

Inputs: AV MIB
Local Configuration Information

Outputs: Local AV

Functionality:

The Select Correct AV function checks its AV Management Information Base (MIB) to obtain the AV which will be used to transfer further access control information to the SDNS component with which it is attempting to establish an association. In the case of an SDNS accountable entity having multiple AVs available to it, the Local Configuration Information will be used to make the determination as to which AV is the correct one to use. Once the correct Local AV has been identified it will be transferred to the KMAP for inclusion in the Attribute Negotiations service. If a usable AV cannot be identified then an audit message may be generated and the control of the processing will be returned to the KMAP.

4.1.11 *ADDITIONAL ACCESS CONTROL INFORMATION*

Inputs: Remote Auxiliary Vector
Selected Security Service Options
Local Configuration Information

Outputs: Process_AV = True/False
Sec_Serv_Opt_Selected = True/False

Functionality:

The Additional Access Control Information function is the point at which any additional access control information and security service options enter the PAA process. If the Local Configuration Information indicates that the additional access control information is required, and a Remote AV has

been received from the KMAP, then Process_AV = True. If either the Local Configuration Information indicates that no further access control information is required and/or no Remote AV has been sent from the KMAP, then Process_AV = False. If the Local Configuration Information indicates that security service options are to be applied to the EV and an agreed upon set has been received from the KMAP, then Sec_Serv_Opt_Selected = True. If no selected security service options have been received from the KMAP, then Sec_Serv_Opt_Selected = False. If Process_AV = True then the Decrypt AV function is called to perform its processing. If Process_AV = False and Sec_Serv_Opt_Selected = True then the Security Service Option Controls function is called to perform its processing. If both Process_AV and Sec_Serv_Opt_Selected are False then the Exit PAA function is called to perform its processing.

4.1.12 *DECRYPT AUXILIARY VECTOR*

Inputs: Encrypted Remote Auxiliary Vector
AV Signature Universal

Outputs: Remote Auxiliary Vector
Decrypt_AV = Pass/Fail

Functionality:

The Decrypt AV function will use the AV Signature Universal to decrypt the remote peer's AV. If the AV is successfully decrypted then Decrypt_AV = Pass and the decrypted remote peer's AV (hereafter referred to as the Remote AV) is passed to the Check AV Signature function. If the AV cannot be decrypted then Decrypt_AV = Fail, and an audit message may be generated and the control of the processing will be returned to the KMAP.

4.1.13 *CHECK AV SIGNATURE*

Inputs: Remote Auxiliary Vector

Outputs: AV_Signature = Pass/Fail

Functionality:

The Check AV Signature function will perform an integrity check on the remote AV to insure that the contents haven't been tampered with. If the integrity check passes then AV_Signature = Pass and the Parse AV Information function is called to perform its processing. If the integrity check fails then

AV_Signature = Fail, an audit message may be generated and the control of the processing is returned to the KMAP.

4.1.14 *PARSE AUXILIARY VECTOR INFORMATION*

Inputs: Remote Auxiliary Vector
Local Configuration Information

Outputs: Parsed Remote AV's Access Control Information
Parse_AV = Pass/Fail

Functionality:

The Parse Auxiliary Vector Information function will use the Local Configuration Information to parse the remote AV and locate all the access control information to be used in later checks. If the required access control information is successfully parsed then Parse_AV = Pass and the Local Rule Based Access Control Checks function is called to perform its processing. If the parsing operation is unsuccessful then Parse_AV = Fail, an audit message may be generated and the control of the processing will be returned to the KMAP.

4.1.15 *LOCAL RULE BASED ACCESS CONTROL CHECKS*

Inputs: Local AV LRBAC Information
Remote AV LRBAC Information
Local Configuration Information

Outputs: LRBAC_Enforced = True/False
LRBAC_PAA = Pass/Fail
Entry in EV

Functionality:

The Local Rule Based Access Control (LRBAC) Checks function performs two tasks: it determines if the LRBAC tier is being enforced by the local SDNS component; and then, if the LRBAC tier is being enforced , it performs the LRBAC PAA checks. These tasks are defined in the following two sections (4.1.15.1 & 4.1.15.2).

4.1.15.1 *LRBAC Enforced Check*

Inputs: Local Configuration Information

Outputs: LRBAC_Enforced = True/False
Entry in Ev

Functionality:

The LRBAC Enforced Check determines if the LRBAC tier is being enforced by the local SDNS component. This function will examine the Local Configuration Information to see if the LRBAC tier is to be enforced. If the Local Configuration Information indicates that the LRBAC tier is to be enforced, then LRBAC_Enforced = True and the LRBAC PAA Check function is called to perform its processing. If the Local Configuration Information indicates that the LRBAC tier is not being enforced, then LRBAC_Enforced = False and the Additional IBAC Checks function is called to perform its processing. When the LRBAC_Enforced = False the EV is set to indicate that the LRBAC tier is not being enforced.

4.1.15.2 *LRBAC PAA Check*

Inputs: LRBAC_Enforced
Local AV LRBAC Information
Remote AV LRBAC Information

Outputs: LRBAC_PAA = Pass/Fail
Entry in EV

Functionality:

The LRBAC PAA Check is responsible for evaluating the Local and Remote AV's LRBAC Information. (Note: For this function to be performed the LRBAC_Enforced boolean must be in the "True" state.) This evaluation will determine if there is any LRBAC information (e.g., attributes, permissions) which is shared by both peers. If there are any LRBAC attributes which both peers hold in common, then LRBAC_PAA = Pass and all of the appropriate LRBAC attributes (i.e., those which will be represented in a PDU's attributes and checked against in PAE) which are held in common between the peers are placed in the EV. Once the applicable LRBAC information is placed in the EV the Additional IBAC Checks function will be called to perform its processing. If there are no LRBAC attributes which are held in common by both peers, then LRBAC_PAA = Fail, an audit message may be generated and the control of the processing will be returned to the KMAP.

4.1.16 *ADDITIONAL IDENTITY BASED ACCESS CONTROL CHECKS*

Inputs: Remote AV Additional IBAC Information
Local Configuration Information

Outputs: AV_IBAC_Enforced = True/False
AV_IBAC_PAA = Pass/Fail
Entry in EV

Functionality:

The Additional Identity Based Access Control (IBAC) Checks function performs two tasks: it determines if the additional IBAC tier information is being enforced by the local SDNS component; and then, if the additional IBAC information is being enforced, it performs the Additional IBAC PAA checks. These tasks are defined in the following two sections (4.1.16.1 & 4.1.16.2). Additional IBAC tier information is any IBAC information which is not included in the set of minimum essential required IBAC information listed for inclusion in the Primary Certificate. The IBAC tier checks are broken into IBAC Checks and Additional IBAC Checks because the IBAC tier is the only tier which can have information contained in both the primary certificate and the AV.

4.1.16.1 *Additional IBAC Enforced Check*

Inputs: Local Configuration Information

Outputs: AV_IBAC_Enforced = True/False
Entry in EV

Functionality:

The Additional IBAC Enforced Check determines if the additional IBAC tier information is being enforced by the local SDNS component. This function will examine the Local Configuration Information to see if the additional IBAC tier information is to be enforced. If the Local Configuration Information indicates that the additional IBAC tier information is to be enforced, then AV_IBAC_Enforced = True and the Additional IBAC PAA Check function is called to perform its processing. If the Local Configuration Information indicates that the additional IBAC tier information is not being enforced, then AV_IBAC_Enforced = False and the Security Service Option Controls function is called to perform its processing. When the AV_IBAC_Enforced = False the EV is set to indicate that the additional IBAC tier information is not being enforced.

4.1.16.2 *Additional IBAC PAA Check*

Inputs: AV_IBAC_Enforced
Remote AV IBAC Information
Local Configuration Information

Outputs: AV_IBAC_PAA = Pass/Fail
Entry in EV

Functionality:

The Additional IBAC PAA Check is responsible for evaluating the Remote AV IBAC Information. (Note: For this function to be performed the AV_IBAC_Enforced boolean must be in the "True" state.) The evaluation will determine if the Remote AV Information passes the Additional IBAC checks which are required by the policy of the local SDNS accountable entity. The exact nature of these Additional IBAC checks will be contained in the local SDNS component's Local Configuration Information.

If the Additional IBAC checks are successfully completed (AV_IBAC_PAA = Pass), any appropriate Additional IBAC information needed for use in PAE will be placed in the EV. Once the appropriate Additional IBAC information is placed in the EV the Security Service Option Controls function will be called to perform its processing. If any of the Additional IBAC checks are unsuccessful, then AV_IBAC_PAA = Fail, an audit message may be generated, and the control of the processing will be returned to the KMAP.

4.1.17 *SECURITY SERVICE OPTION CONTROLS*

Inputs: Selected Option Set
Enforcement Vector

Outputs: Enforcement Vector
EV_Valid = True/False

Functionality:

The Security Service Option Controls function will use the agreed upon set of security service options from the KMAP to possibly refine the information already contained in the EV. An example of a possible refinement to the EV might be the reduction of an allowable set of IPSO specified field values

down to a single IPSO field. The following options are available for selection: confidentiality, integrity, confidentiality and integrity, per PDU security label and format, single security level, and final sequence numbers. The only options which may have a bearing on the EV are the single security level option, and the per PDU security label and format option.

The applicable security service options are applied to the EV. If the application of the security service options to the EV results in a refined and valid EV, the refined EV will replace the old one and EV_Valid = True. If no refinements are done to the EV then it will remain unchanged and EV_Valid = True. If the refinements to the EV result in an invalid EV then EV_Valid = False. The Exit PAA function will then be called to perform its processing.

4.1.18 *EXIT PAA*

Inputs: EV_Valid

Outputs: PAA_Result = Pass/Fail
EV

Functionality:

The Exit PAA function will determine a PAA result (PAA_Result = True/False) and forward that result to the KMAP. If EV_Valid = True then PAA_Result = Pass, PAA_Result and the EV will be transferred to the KMAP indicating the success of PAA. If EV_Valid = False then PAA_Result = Fail, an audit message may be generated, and the PAA_Result will be transferred to the KMAP indicating the failure of PAA.

4.2 Peer Access Enforcement

The Peer Access Enforcement (PAE) process provides for the continuous enforcement of the access control decision made in the PAA process. PAE determines if the security sensitivity of a data packet, represented in a PDU's attributes, falls within an allowable set of security attributes determined during PAA. The enforcement mechanism becomes involved when peer Security Protocols (SP) attempt to transfer data. The full suite of PAE processing is only required when the label contained on a specific piece of data (either imported or exported) is an explicit label. When implicit labeling is being used, only a limited subset of PAE is used, the PAA process provides sufficient access control assurance.

The PAE process is called from the SP to perform its functionality. The SP acts as a gate keeper for all data packets, both incoming and outgoing. When the SP receives a packet for processing it checks to

see if the appropriate pointer is in place (a TEK is used when sending and receiving a packet, the KID may be used but this is only a pointer to the TEK). If this pointer is not in place the SP will enter an error condition and drop the packet (some implementations may allow for an optional recovery attempt to be made, the packet being processed may be cached with the processing being transferred to KMP in an attempt to establish an association). At the appropriate time in the SP processing the PAE process will be called to perform its processing. Figure 4.3 illustrates the functions which the PAE process will perform. The PAE process will yield a result (PAE_Result = Pass/Fail) which will be returned to the SP. It is this result which will determine the remainder of the processing which the SP must perform. The following sections describe the details of the PAE process.

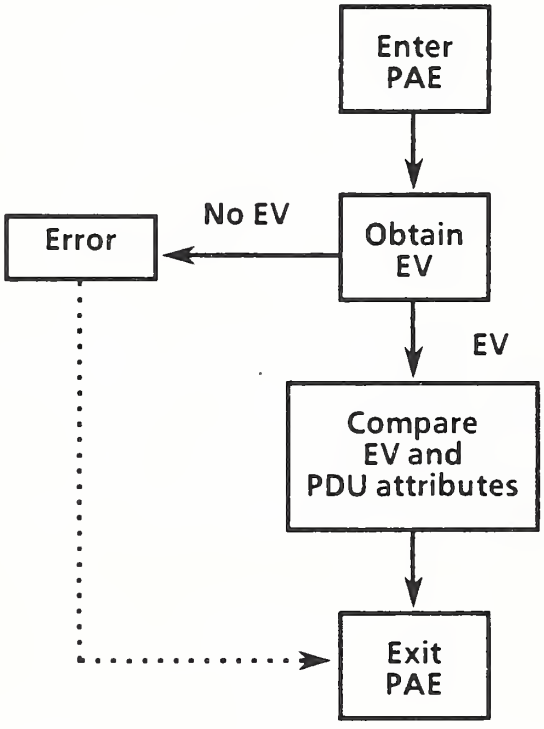


Figure 4.3. Peer Access Enforcement (PAE) Process

4.2.1 *ENTER PAE*

Inputs: Request from SP to Perform Access Control Enforcement Processing

Outputs: Request to Obtain Enforcement Vector
PDU Attributes

Functionality:

The Enter PAE function will be called by the SP when access control enforcement processing is to be performed on a PDU's attributes for a data packet. This function will get the PDU's attributes for the data packet being processed from the SP. The PDU's attributes will be transferred, along with a request, to the Obtain Enforcement Vector function so that it can perform its processing.

4.2.2 *OBTAIN ENFORCEMENT VECTOR*

Inputs: Request to Obtain Enforcement Vector
PDU Attributes

Outputs: EV_Found = Yes/No
Enforcement Vector
PDU Attributes
Error Condition

Functionality:

The Obtain EV function will search the available EV MIB in an attempt to find the correct EV for the cryptographic association which is being used. This function will use the appropriate information contained in the PDU's attributes (e.g., KID, TEK identifier) to try and find a valid EV. If a valid EV is found then EV_Found = Yes, and the EV and PDU's attributes are sent to the Compare EV and PDU Attributes function so that it can perform its processing. If a valid EV is not found then EV_Found = No, and an error condition occurs. This error condition will drop the packet and return processing back to the SP (some implementations may allow for an optional recovery attempt to be made, the packet being processed may be cached with the processing being transferred to KMP in an attempt to establish an association).

4.2.3 *COMPARE EV AND PDU LABEL*

Inputs: Enforcement Vector
PDU Attributes
Local Configuration Information

Outputs: PAE_Result = Pass/Fail

Functionality:
The Compare EV and PDU Attributes function will determine whether or not the security attributes of a data packet, contained in the PDU's attributes, fall within the bounds specified in the EV. This function uses the local configuration information to determine the interpretation of the PDU's attributes appropriate for comparing with the EV. Once interpreted, the PDU's attributes are compared to the EV. If the fields in the PDU's attributes fall within the bounds contained in the EV, then PAE_Result = Pass. If the fields in the PDU's attributes exceed the bounds contained in the EV, then PAE_Result = Fail. The Exit PAE function is then called to perform its processing.

4.2.4 *EXIT PAE*

Inputs: PAE_Result

Outputs: Response to SP containing PAE_Result

Functionality:
The Exit PAE function will, after a PAE result has been determined, return the control of the processing to the SP function which called for the PAE processing. PAE_Result is also passed back to the SP as a response, indicating the success or failure of the PAE operation.

4.3 Staged Delivery Communications Services; Secure Messaging

This section describes the application of the access control model to a store-and-forward communications context. It applies the model to electronic messaging (the only store-and-forward communications service currently being addressed by SDNS). Familiarity with the SDNS Message Security Protocol (MSP) document (SDN.701), is assumed. Figure 4.4 illustrates a functional flow diagram for the order in which access control events occur when called by a User Agent (UA) implementing the MSP.

The following assumptions have been made to better illustrate the application of access control in the context of secure messaging.

- The access control function is integrated within the SDNS MSP implementation, which is also responsible for controlling the management operations. For the case of X.420 interpersonal messaging, MSP operates in conjunction with the UA process responsible for processing mail.

- Given the first assumption, it is technically feasible to reflect information resulting from the access control decision process to the user associated with the UA. (This contrasts with the front

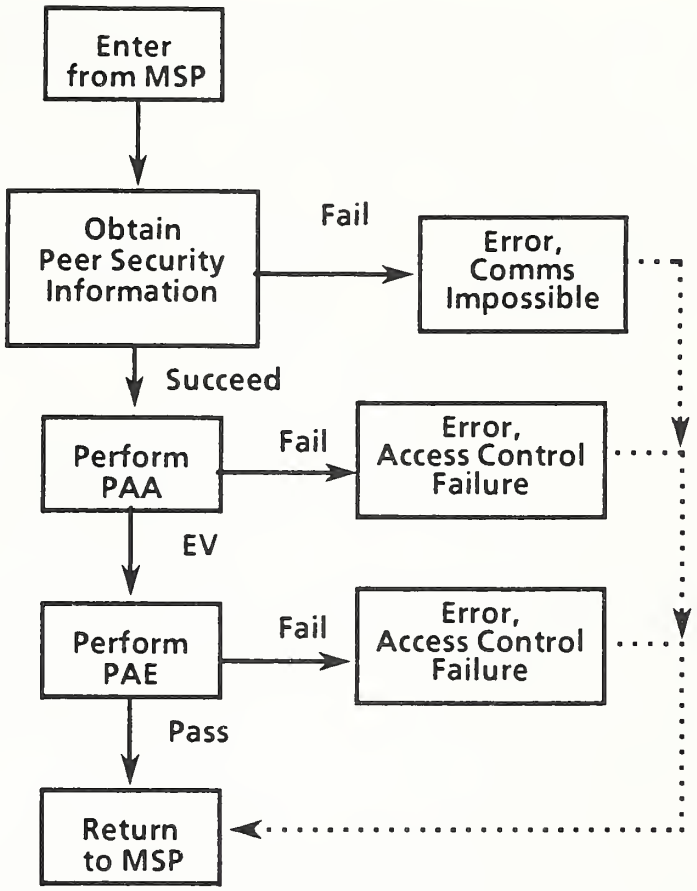


Figure 4.4.Staged Delivery Access Control Services; Secure Messaging

end SDNS implementation often contemplated at layers 3 and 4, where a protocol between the associated host and its SDNS front end may or may not exist.) Reflection of such data may be restricted under the local policies to be implemented in some SDNS components.

- In environments where the use of AVs is necessary, appropriate AV representations will be accommodated within directory service records and message OriginatorSecurityData structures.

- For purposes of this description, it is assumed that access control enforcement is configured within originator and recipient MSP implementations as a required service, and is applied to all outbound and incoming message traffic.

- It is assumed that a Composite ID represents a user's privilege attributes. If a user's operating range must be constrained based on characteristics of the associated UA or its supporting system, application of such constraints is a local matter outside the scope of SDNS access control.

In response to a user's request, an originator UA submits an outgoing message for MSP processing. Message security label information is determined in one of two ways: (implicitly) based on local processing context, or (explicitly) based on the SecurityLabel parameter included in the associated submission envelope. Submission envelope parameters processed by MSP (in addition to message content itself) include:

- SecurityLabel
- recipient O/R name list
- OriginatorIdentifier (if not implicit)
- ContentType designator for message

If multiple recipients are specified, access control checks (including calls to the PAA and PAE procedures) are performed for each recipient. A distinct EV governs message transfer to each recipient of a multicast message. If access control checks fail for some recipients of a multicast message, this need not preclude transmission of the message to those recipients for whom the checks succeed. An originator may (depending on local implementation and policy) be informed when one or more of a message's recipients are unacceptable for access control reasons. Optionally, an originator may request that MSP processing and message transmission be completed only if all listed recipients are acceptable.

If access control processing is successfully completed, MSP encryption processing is performed. The remainder of the section details the access control functions defined in figure 4.4, as called by MSP.

4.3.1 *ENTER FROM MESSAGE SECURITY PROTOCOL (MSP)*

Inputs: Request from MSP to Perform Access Control Processing

Outputs: Request to Obtain Peer Security Information

Functionality:
The Enter From MSP function will be called by the MSP when access control processing is to be performed on a message, either incoming or outgoing. Upon receipt of a request to perform access control processing from MSP, a request is sent to the Obtain Peer Security Information function to perform its processing.

4.3.2 *OBTAIN PEER SECURITY INFORMATION*

Inputs: Request to Obtain Peer Security Information
Local Configuration Information

Outputs: Obtain_Peer_Info = Success/Failure
Remote Composite ID Information
Error Condition

Functionality:
The Obtain Peer Security Information function is responsible for obtaining the remote peer's Composite ID. This function is further broken down into obtaining the information in both the outgoing and incoming situations (4.3.2.1 & 4.3.2.2).

4.3.2.1 *On Outgoing Messages*

Inputs & Outputs are the same as for the Obtain Peer Security Information function above.

Functionality:
The Obtain Peer Security Information On Outgoing Messages function is responsible for obtaining the message recipient's posted certificate, RandomPart, and AV (if used) from MSP (as an example it is MSP which will perform any directory queries which may be needed). The exact method for obtaining the message recipient's information and the medium on which the message recipient's information is conveyed is a local matter and will be determined through the application of local configuration information. (e.g., Recipient information may be posted on a directory server, in a local cache, or in a remote cache) If the attempt to obtain the message recipient information is successful, then Obtain_Peer_Info = Success and the message recipient's information is sent to the Perform PAA function so that it can perform its processing.

If the attempt to obtain the message recipient's information is unsuccessful, then Obtain_Peer_Info = Failure and communications are impossible. The failure to obtain the message recipient's information results in the generation of an error condition, this error condition should not be construed as an access control decision but rather is a communications failure. Control of the processing is returned to MSP. In the case of a multicast message this error condition is only tied to the specific recipient for whom the attempt to retrieve posted information was being made.

4.3.2.2 *On Incoming Messages*

Inputs & Outputs are the same as for the Obtain Peer Security Information function above.

Functionality:

The Obtain Peer Security Information On Incoming Messages function is responsible for obtaining the message's originator-specific information. All such information is available in the OriginatorSecurityData within the message's security heading. If the attempt to obtain the message's originator-specific information is successful, then Obtain_Peer_Info = Success and the message's originator-specific information is sent to the Perform PAA function so that it can perform its processing.

If the attempt to obtain the message's originator-specific information is unsuccessful, then Obtain_Peer_Info = Failure and communications are impossible. The failure to obtain the message's originator-specific information results in the generation of an error condition, this error condition should not be construed as an access control decision but rather is a communications failure. Control of the processing is returned to MSP. Note that this case can only occur upon receipt of a defective message (i.e., one which doesn't contain all of the MSP-required contents).

4.3.3 *PERFORM PAA*

Inputs: Message Originator's Composite ID Information
Message Recipient's Composite ID Information

Outputs: PAA_Result = Pass/Fail
Enforcement Vector (EV)
Error Condition

Functionality:

The purpose of the Perform PAA function is to compare both the message originator's and message recipient's Composite ID information to determine whether or not a commonly held set of security attributes exists on which an association can be based. In the PAA procedure, a UA is responsible for calling all the PAA functionality which the local configuration dictates. For a detailed description of PAA refer to section 4.1.

If PAA is successfully completed then PAA_Result = Pass, and the EV is passed to the Perform PAE function so that it can perform its processing. If the PAA is unsuccessful then PAA_Result = Fail and an error condition is generated. The error condition generated by a failed PAA would indicate that no allowable set of communications exists between the two peers, this is an access control decision. In the case of an outgoing multicast message, this error condition is only tied to the specific recipient for

whom the attempt to perform PAA is made. If PAA fails, control of the processing is returned to the MSP along with the appropriate error condition.

4.3.4 *PERFORM PAE*

Inputs: Message Sensitivity Label
EV

Outputs: PAE_Result = Pass/Fail

Functionality:
The Perform PAE function determines whether a message meets the constraints represented in an EV. At an originator UA, RBAC determinations are based on submission envelope fields representing security-relevant attributes and/or on security attributes determined implicitly based on processing context. At a recipient UA, RBAC determinations are based on the sensitivity indication carried within the ProtectedToken carried in the message and corresponding to that recipient.

It is assumed that IBAC enforcement at an originator is based on the set of recipient Originator/Recipient (O/R) names to whom a message is directed and, correspondingly, that IBAC enforcement at a recipient is based on the originator O/R name as extracted from the originatorCertificate in an incoming message. If the IBAC checks will be performed in the course of PAA processing (as EV establishment is attempted) they need not be incorporated within PAE.

If the PAE function is successfully completed then PAE_Result = Pass, and the Return to MSP function is called to perform its processing. If the PAE function is unsuccessful then PAE_Result = Fail, and an error condition is generated. The error condition generated by a failed PAE would indicate that the message being examined did not fall within the bounds established for that communications during PAA, this is an access control decision. In the case of an outgoing multicast message, this error condition is only tied to the specific recipient for whom the attempt to perform PAE is made. If PAE fails, control of the processing is returned to the MSP along with the appropriate error condition.

4.3.5 *RETURN TO MSP*

Inputs: PAE_Result

Outputs: Response to MSP indicating successful access control processing

Functionality:

The Return to MSP function will return control of the processing to the MSP function. Along with the return of control this function will also indicate to the MSP that the message which was being processed has successfully completed all access control procedures.

5.0 ACCESS CONTROL INFORMATION SPECIFICATION (ACIS) OVERVIEW

5.1 Introduction

The requirement for the Access Control Information Specification has been established as a result of the Access Control Working Group's (ACWG) investigation of various Partitions. This tool may be applied to all Partitions, but as yet a detailed investigation of the Security Attribute structures of all Partitions has not been made.

Investigations have shown that the Security Attributes of many environments are both hierarchical and non-hierarchical in nature and can be represented as a tree structure. This organizational scheme posed some problems when trying to represent this data in a Primary Certificate or in an Auxiliary Vector. If the data is broken down into tiers and an entire tier is represented contiguously before moving to the next tier, ambiguities may arise when trying to associate lower tier attributes with the current tier being examined. On the other hand representing each branch of the tree completely and independently requires too much space. More importantly, it has been found that without some uniform method for encoding and comparing access control data many SDNS implementations would require software specific to their local security policy. With this set of issues to be resolved, the ACWG set out to develop the Access Control Information Specification (ACIS).

ACIS is a tool to be supported by vendors who develop SDNS components supporting access control and to be used by SDNS customers. ACIS provides several useful functions:

(a) It provides an approach to enforcing access control which is independent of any particular security policy, leading to a savings in development costs for an implementation;

(b) It provides a uniform method for representing access control information through the use of its grammar;

(c) It provides a standard algorithm for interpreting and comparing access control attributes.

ACIS is based upon the theory of predictive parsing. The grammar is a form of an LL(1) grammar. A more detailed description on ACIS can be found in the SDNS Access Control Specification, Addendum 1 (SDN.802/1), the Access Control Information Specification.

5.1.1. *USER REQUIREMENTS FOR ACIS*

ACIS is designed to handle problems that arise when the Security Attributes of a particular system have certain characteristics. The most important of these characteristics is that the system Security Attributes can be expressed in a tree structure, implying hierarchy and interdependence. A second characteristic of the system is that its access control must be able to be performed as an intersection process of the attributes of the two entities that wish to communicate.

Consistent with these characteristics, the first task to accomplish when using ACIS is to express the entire System Security Attribute structure of a particular Partition as a tree structure. Appended to the System Security Attribute tree should be a collection of possible PDU attributes which indicate what label may be expected if communications is allowed for this set of attributes. Figures 5.1 and 5.2 show a system tree and a set of allowable labels, respectively.

Within a Partition each communicating entity might have a subset of the overall System Security Attributes. A partial tree must be developed for each of these entities and it must be consistent with the system tree. Furthermore, the expressions (or partial trees) for any two entities that wish to communicate must have the exact same interpretation of each of the branches on which communications will be allowed. Figures 5.3 and 5.4 show a sample tree for Hosts A and B which has been derived from the system tree of Figure 5.1.

This preliminary organization and assignment of access control attributes is an essential part of making ACIS work and must therefore be done correctly to insure the accurate performance of access control. For the most part this work is consistent with preliminary tasks required today for systems employing access control, and therefore should not introduce a significant amount of new work.

There are two cases to consider when deciding what portions of the System Security Attributes tree go into the Primary Certificate and/or the Auxiliary Vector. It should be noted that ACIS will not be used on any information placed in the primary certificate. Any interrelationships between any of the fields will be represented in the Auxiliary Vector. The first case, which is primarily the one addressed above, is when there is a single tree for the entire collection of attributes spanning all four tiers of the Four-Tiered Model. In this case, if all of the information cannot fit into the primary certificate a means for transferring or duplicating some primary certificate information may need to be done to complete the tree in the Auxiliary Vector. The second case to consider is when each set of the attributes for a particular tier in the model is a separate tree in and of itself. In this case, if the information cannot fit into the primary certificate then the information must be split between primary certificate and auxiliary vector. The information would be broken down, the Tier 1, Tier 2, and some Tier 4

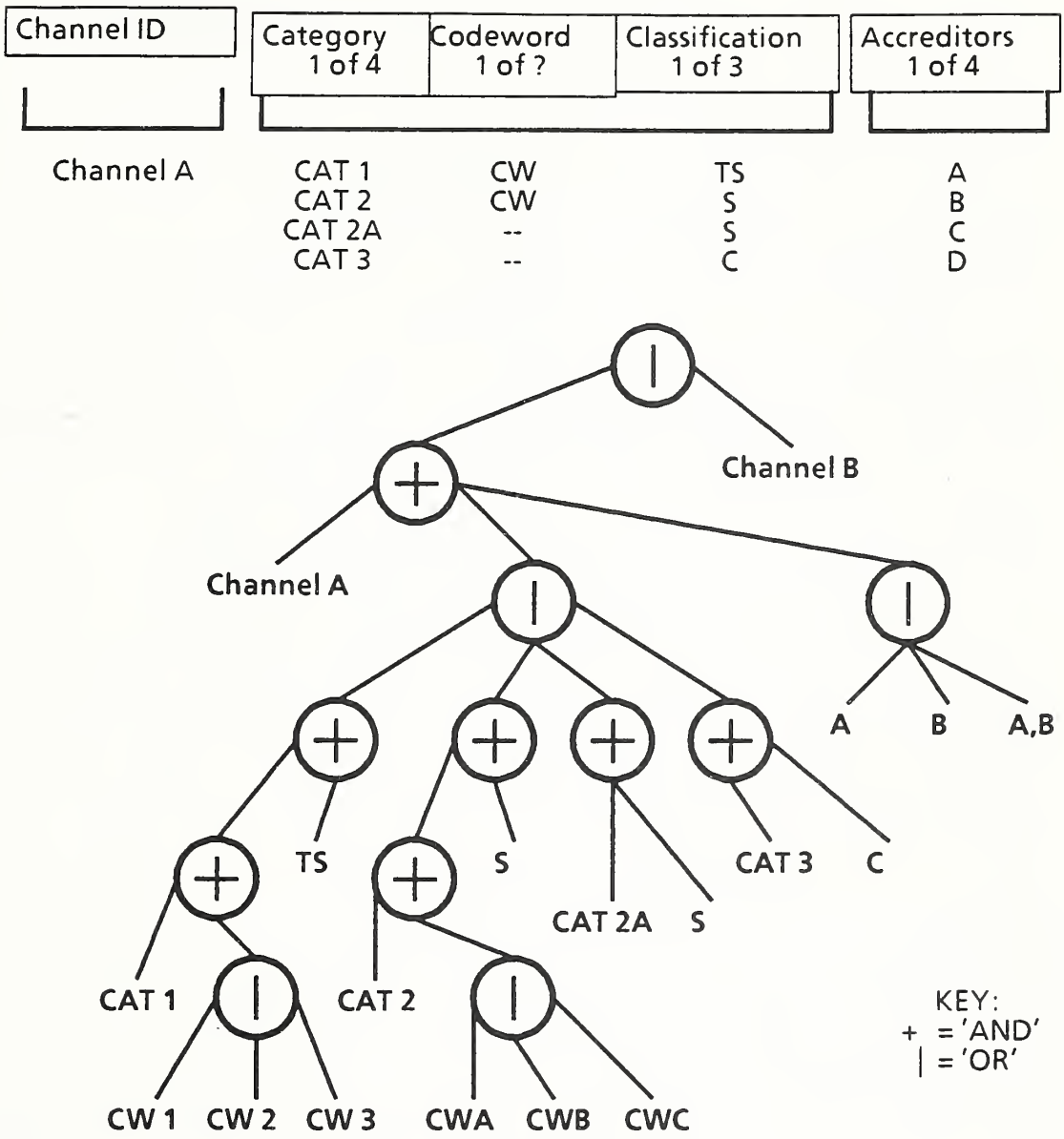


Figure 5.1. System Security Attribute Tree

information will be in the Primary Certificate and the Tier 3 and 4 information will be in the Auxiliary Vector.

5.1.2 *COMPONENTS OF ACIS AND HOW IT WORKS*

ACIS is constructed of several functions and a grammar. As seen in the previous figures, logical operators are required in order to accurately express the interrelationships of Security Attributes. Shown in the figures (for the purpose of example) are the 'and' and 'or' operators. Other operators are

```
CHANNEL A LABEL ; = CHANNEL_ID + SECURITY_ID + ACCREDITOR

CHANNEL_ID : = 'CHANNEL A'

SECURITY_ID : = CAT1 | CAT2 | CAT2A | CAT3

CAT1 : = CAT1_ID + CAT1_CW + TS_ID

CAT1_ID : = 'CAT1'

CAT1_CW : = 'CW1' | 'CW2' | 'CW3'

TS_ID : = 'TS'

CAT2 : = CAT2_ID + CAT2_CW + S_ID

CAT2_ID : = 'CAT2'

CAT2_CW : = 'CWA' | 'CWB' | 'CWC'

S_ID : = 'S'

CAT2A : = CAT2A_ID + S_ID

CAT2A_ID : = 'CAT2A'

CAT3 : = CAT3_ID + C_ID

C_ID : = 'C'

    ACCREDITOR : = 'A' | 'B' | 'A,B'
```

Figure 5.2. Channel A Allowable Label Set

detailed in the SDNS Access Control Specification, Addendum 1 (SDN.802/1); the ACIS Detailed Specification. Associated with each operator is a unique character. This set of operators and their representations comprise the ACIS grammar. This grammar is used when constructing the System or Entity Attribute tree.

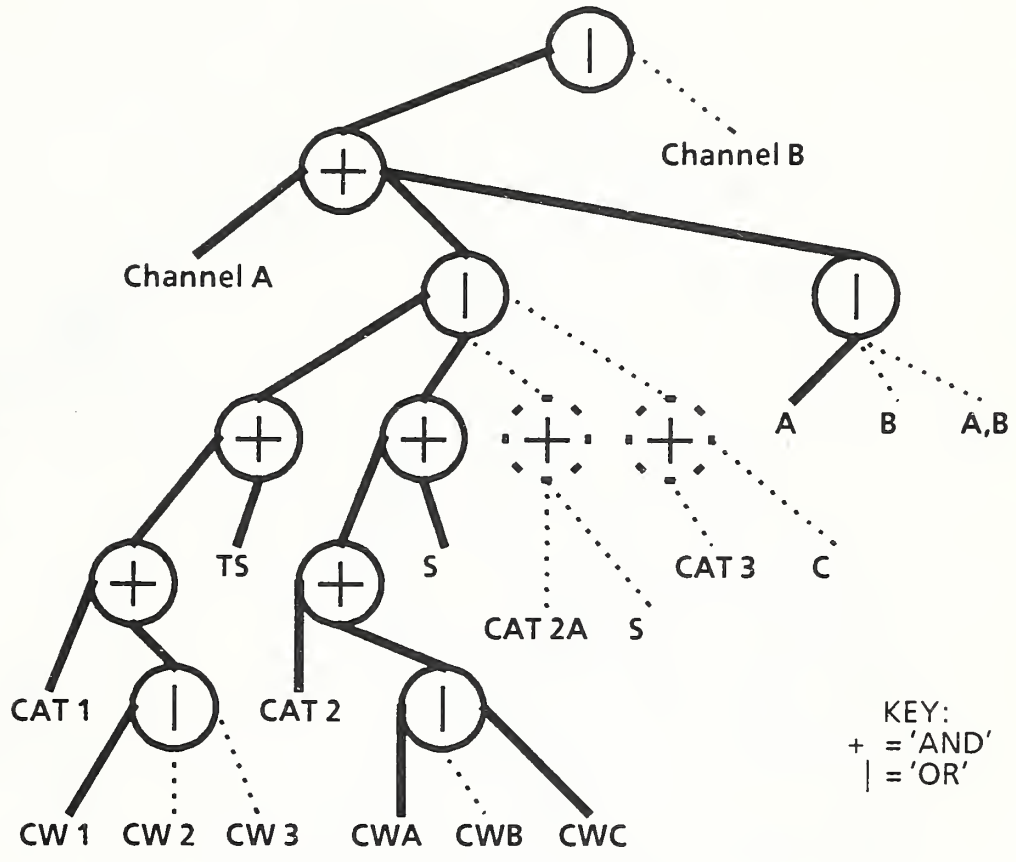


Figure 5.3. HOST A's Permission Set

The first of several functions of ACIS is the encoding function. The process takes the tree structured attributes as an input and writes it as a set of contiguous characters which can then be placed inside an auxiliary vector. This string is called an Access Control Expression. Figure 5.5 shows the encoding process of the Entity we called Host A. Conversely, there is a decoding function which takes as input a contiguous string of bytes and can reconstruct the System or Entity tree. This concept is shown in Figure 5.6.

During the PAA process the Security Attributes of the entities wishing to communicate are exchanged. In order to determine if these two entities have Security Attributes in common, an intersection or 'Comparison' process must take place. The Compare function in ACIS takes as input any two Encoded strings and eliminates the nonintersecting attributes. Areas where the two intersect are preserved as information in the EV for the PAE process. Note that the full suite of PAE is only relevant when explicit labeling is used and when label values may vary during the lifetime of an association. Those portions of either string which do not intersect are eliminated (pruned). In cases

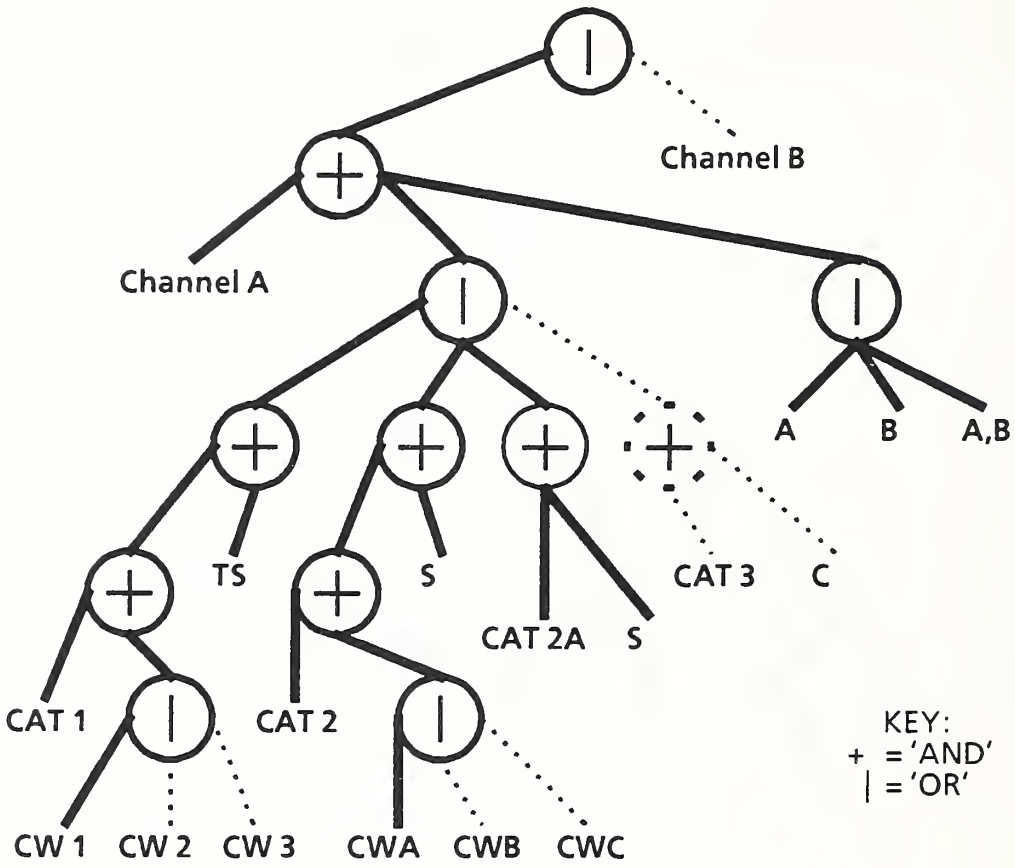
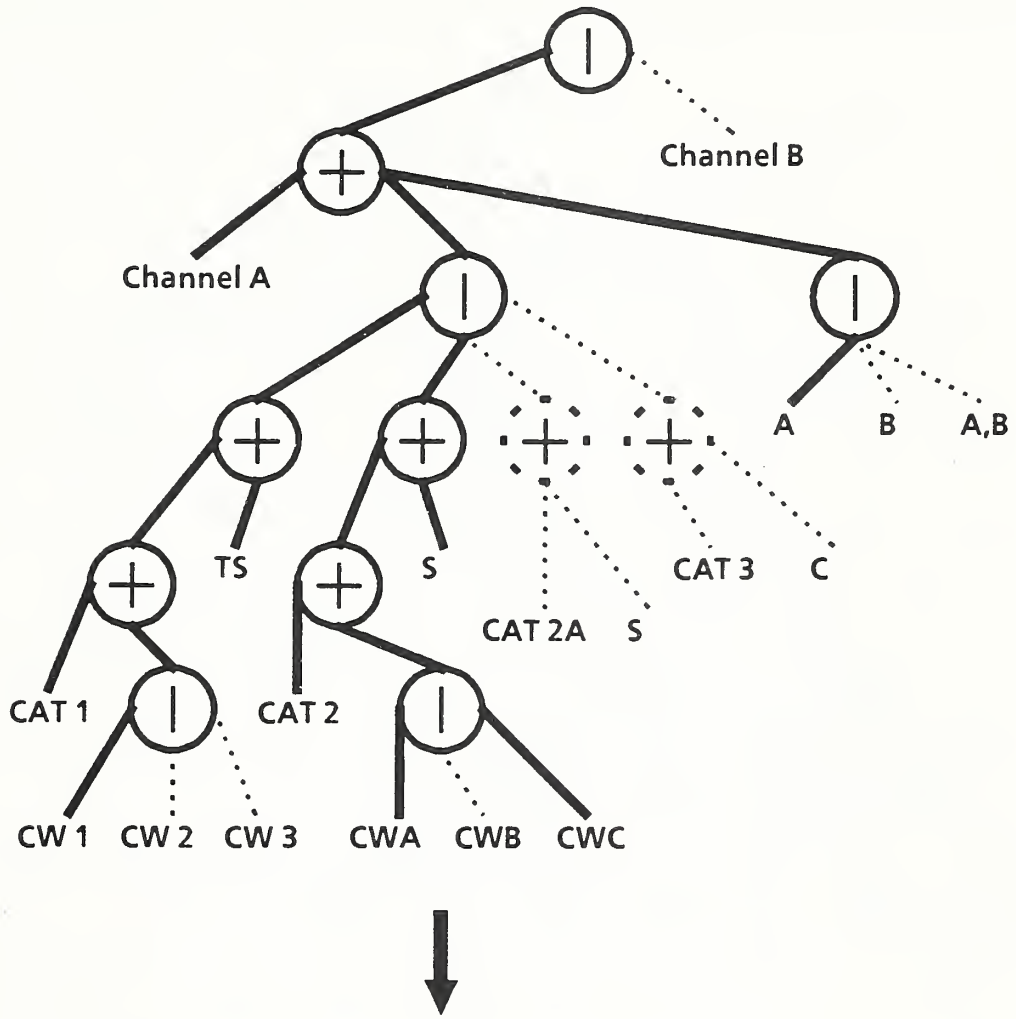


Figure 5.4. HOST B's Permission Set

where no intersection exists the communication is disallowed. This Comparing function of ACIS must assure the elimination of unauthorized labels, as well as guarantee authorized communications. Figure 5.7 conceptually shows this pruning process for Hosts A and B.

If communications are permitted, continuous access control is affected by means of the PAE process. This process involves comparing a PDU's attributes with the contents of an EV. When a path is found that is common to both parties, the end of that path should contain a set of allowable labels representing that path. When PAE is performed, the PDU's attributes are compared with this set extracted from the EV.

The following is an example of a high level scenario. The security officer for a partition will construct the System Security Attributes tree. From this all subsequent Entity trees will be derived. When an Entity's representative wants to enter that Entity into the system, the appropriate paperwork is filled out. At the ordering station/s (two if the information is carried in separate chunks) the information is entered into a computer system which checks and encodes the attributes. The resulting Access Control



((CHANNEL A) ((((CAT1)((CW1);|)(TS); +)(((CAT2)((CWA)(CWC);|); +)(S); +);|)((A);|); +);|)

Figure 5.5. Encoding HOST A's Permission Set

Expression is then entered into the Auxiliary Vector. The attributes are then loaded into the SDNS component and exchanged and checked as per Peer Access Approval and Peer Access Enforcement, discussed in the previous section.

The SDNS Access Control Specification, Addendum 1 (SDN.802/1) specifies the ACIS grammar, the operators, the encoding and decoding functions and the compare function. From this level of detail ACIS implementors should be able to code and operate this tool.

((CHANNEL A) ((((CAT1)((CW1);|)(TS); +)(((CAT2)((CWA)(CWC);|); +)(S); +);|)((A);|); +);|)

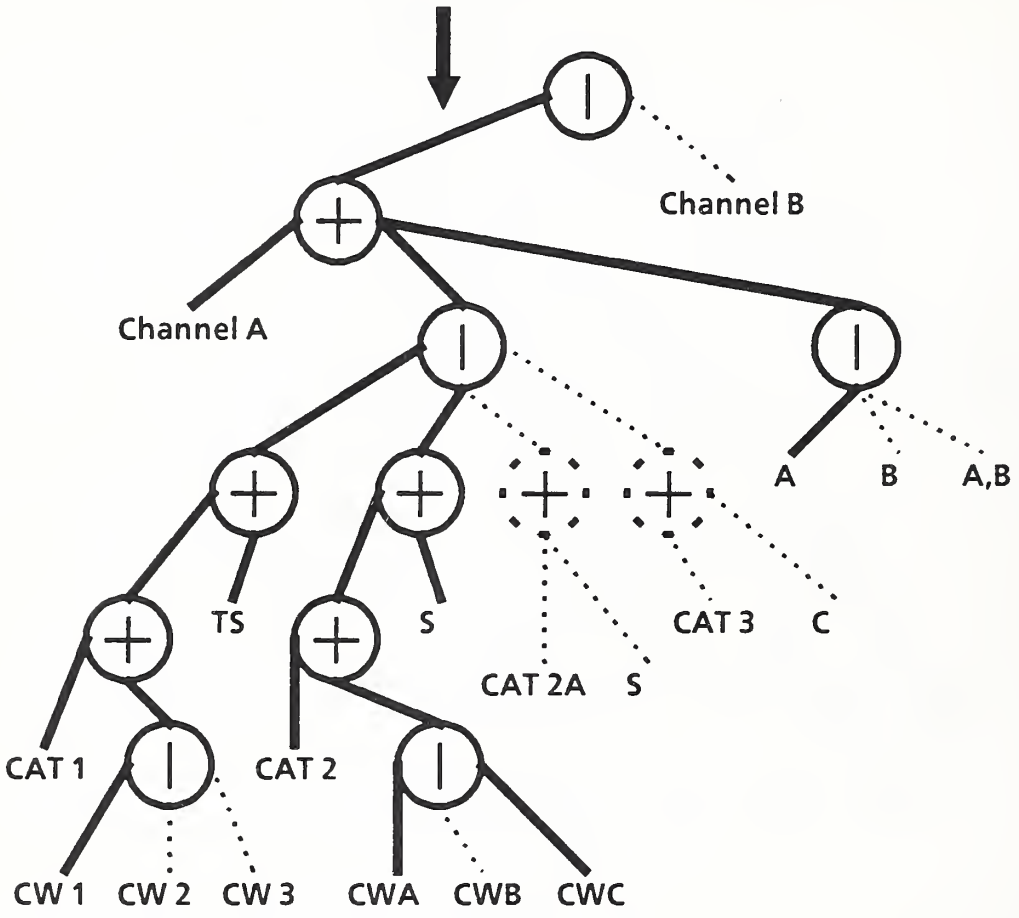


Figure 5.6. Decoding HOST A's Permission Set

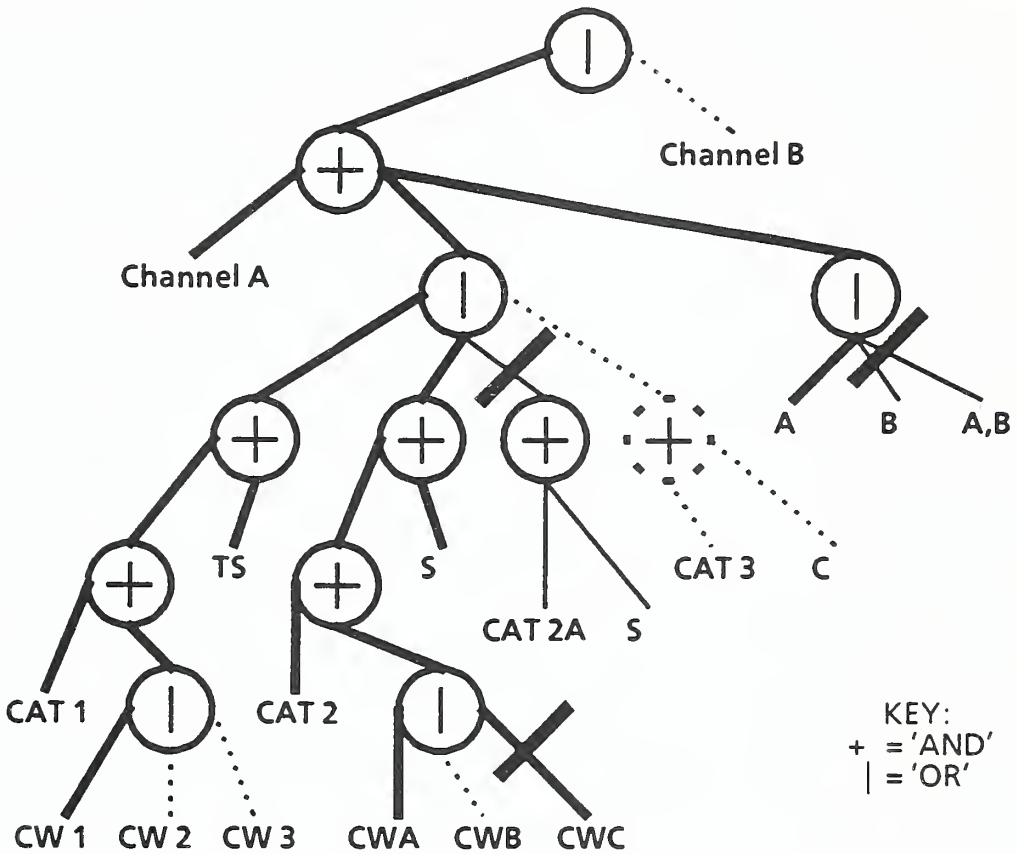


Figure 5.7. HOST A & HOST B Permission Intersection

ACCESS CONTROL SPECIFICATION

ACCESS CONTROL INFORMATION SPECIFICATION (ACIS) ADDENDUM 1 (SDN.802/1)

July 25, 1989

TABLE OF CONTENTS

TABLE OF CONTENTS

ACCESS CONTROL INFORMATION SPECIFICATION (ACIS) ADDENDUM

1.0 INTRODUCTION

This addendum is an extension of the ACIS Specification discussion provided in section 5 of the Access Control Specification (SDN.802). In particular it provides a detailed explanation of the capabilities, limitations, and implementation requirements for ACIS.

Figure 1.0 presents an overview of the SDNS ACIS System architecture. In the top part of the figure, system attributes and entity attributes are inputs to process 1 which runs on the Access Control Attribute definition device. SDNS calls this device a Local Authority Workstation (LAW). This process yields attribute trees for each of the entities within the system (Entity 1 through Entity N). Process 1 is composed of two major functions.

The first function is to take the various system access control attributes and define a complete access control policy which is appropriate for expression in ACIS and for enforcement by SDNS components within the system.

The second part of the function is to represent those portions of the system attributes that are appropriate for each entity in the system. This corresponds in a general sense to the ordering process. Out of this process comes a representation of the pertinent attributes for each entity in the system. This attribute representation is called an entity attribute tree in the figure.

Next these access control attributes and policy representations are exchanged by entities in the system in order to authorize communication between entities and to determine the rules associated with the communication. This process is labelled as process 2. These rules and attributes are contained in the Enforcement Vector (EV).

Finally, PDUs are tested against the EV in process 3 to ensure that they conform with the communication rules established in Process 2.

Section 2.0 discusses definition of the access control policy representation to be enforced by an SDNS component. Included is a description of those elements that go into an access control policy representation and the operators and syntax rules supported by ACIS. Examples of a control system containing PAA type information and PAE

labelling information are also presented. In a later version of this spec both sets of information will be consolidated into a tree in order to represent a complete policy representation example.

Section 3.0 (Encoding of Entity Attributes) discusses the ordering process for entity attributes and the encoding of those attributes.

Once the entity attribute trees are delivered to the SDNS component, they are exchanged between entities in order to authorize communication and to create an Enforcement Vector (EV). SDNS calls this process Peer Access Approval (PAA). It is labelled process 2 in Figure 1. This process is the subject of section 4.0 (Comparing of Access Control Attributes).

Finally process 3 uses the EV created during PAA to test PDU security labels. This ensures communication at only the security levels authorized during PAA. The process is called Peer Access Enforcement (PAE) and is the subject of section 5.

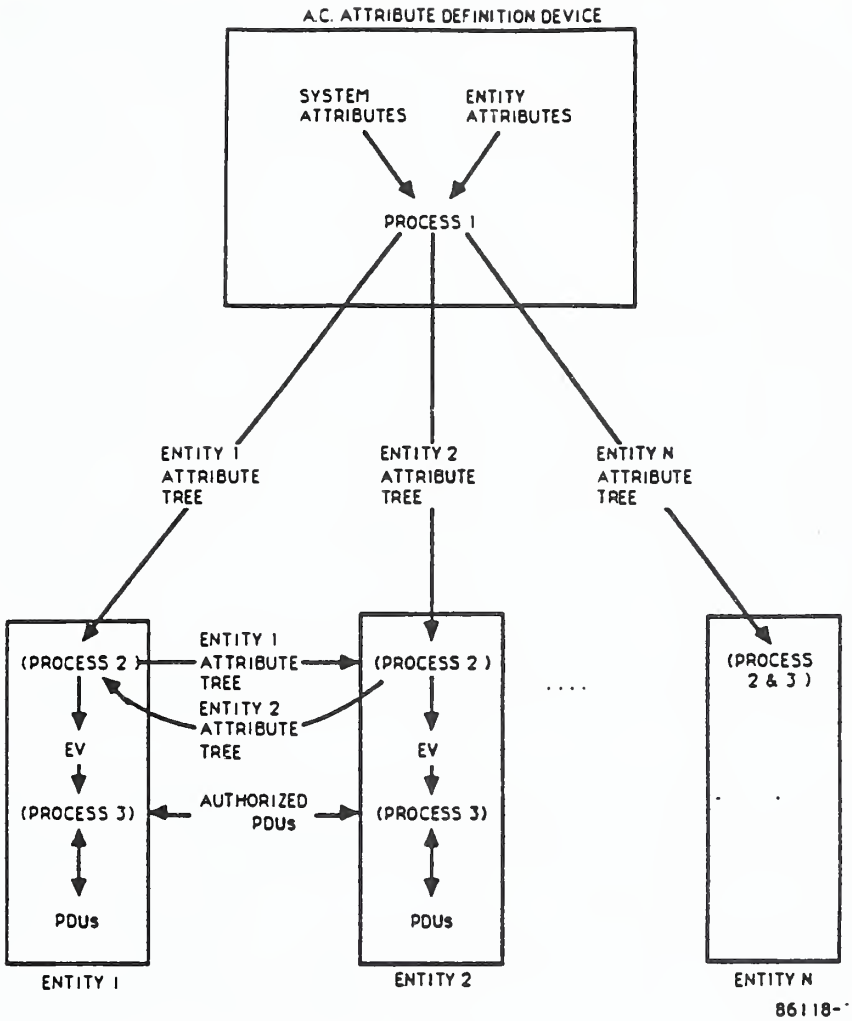


Figure 1 ACIS Overview

2.0 ACCESS CONTROL POLICY REPRESENTATION

Section 2.1 presents a discussion of those elements that must be included in an access control policy representation. Section 2.2 presents the operators and syntax rules for ACIS in definition of an access control policy representation to be enforced by SDNS components. Section 2.3 provides examples of policy representations using the ACIS operators and syntax rules defined earlier.

2.1 Access Control Policy Requirements Definition

This function includes two subfunctions: partition access control

policy requirements definition and local access control policy requirements definition. These functions are similar with one relating to partition wide requirements and the other relating to local authority requirements. Within an access control partition, the representation of partition policy is uniform. The same is not true for local policy since it is possible for a partition to support multiple local access control policies.

These functions are identified separately because the responsibility associated with each could potentially fall to different organizations or sub-organizations. An access control partition may support many different local administrations. Each of these local administrations could potentially use different local access control policies supported by locally administrated ordering equipment. Partition access control attributes are placed in primary certificates by the Key Management System (KMS) while local access control attributes are placed in auxiliary vectors by the Local Authority Workstation.

2.1.1 PARTITION ACCESS CONTROL POLICY REQUIREMENTS DEFINITION

This activity is composed of four sub-functions: Partition RBAC Requirements Definition, Partition Ordering Restrictions Definition, Partition IBAC Requirements Definition and Partition Access Control Ordering Forms Design. These functions are discussed in greater detail in the following paragraphs. Definition of the Partition Access Control Policy Requirements is not an SDNS responsibility. SDNS imposes requirements upon the task in order for the rest of the SDNS Access Control system to operate properly. This function is the responsibility of the subscriber administration.

2.1.1.1 Partition RBAC Requirements Definition

This function is composed of several activities. One of the activities is identifying the various access control attributes that are to be incorporated into the rule-based SDNS component enforcement policy. This task entails identifying those items out of all those which exist in the system to make a part of the SDNS RBAC enforcement. Some of the access control items which are appropriate for access control at a user level, for example, may not be appropriate for use by a network level SDNS device. During the process it is important to keep in mind the restrictions imposed by the SDNS ACIS supported mechanisms. These will be discussed in section 2.2. Identification of the access control items includes defining the allowable values that these items can assume and any

relationships that exist between items. It is possible, that some items may be redundant so that not including them in the SDNS rule-based enforcement will not result in any loss of enforcement functionality.

After the rule-based access control attributes are identified, a grammar is constructed which defines the allowable values for the access control items, their relationships to one another, and the SDNS enforceable rules associated with these items. SDNS imposes certain restrictions on the grammar definition to ensure that the SDNS ACIS mechanism can support the proposed functionality.

Next a grammar consisting of PAA and PAE information is constructed. The PAA portion of the grammar represents access control attributes that are used during the PAA process to authorize communication between two subscriber entities. The PAE grammar represents only those items which are used in PDU labels and which are tested by the enforcement vector during PAE. These portions of the grammar are separately identified in the discussion because the access control attributes that are represented in an enforcement expression to authorize communication between two subscriber entities (PAA) are very likely different than the attributes that are carried in PDU labels after communication has been authorized (PAE). For example, the COMPUSEC Certification level of a subscriber entity may be part of a policy to determine whether two subscriber entities may communicate but would probably not be appropriate to carry in a PDU label after communication is authorized. The PAE labelling information is also quite likely an extension of the PAA information, i.e., the PAA attributes will dictate the appropriate PDU label values.

After the grammar is defined a check is performed to ensure that the various productions in the grammar can be enforced by the SDNS component ACIS processing rules in an unambiguous manner. If ambiguities are uncovered, changes must be made in the grammar to eliminate them and the process repeated.

Finally after an unambiguous grammar for the partition is defined, the final step is to represent the grammar in the form of an ACIS partition tree. This step consists basically of taking the productions in the grammar and representing them in the prescribed form for an ACIS tree.

This function along with the analogous function Define Local Access Control Policy Requirements corresponds to blocks 1 and 2 of Figure 2.

2.1.1.2 Partition Ordering Restrictions Definition

This task consists of determining any restrictions that may apply to the SDNS access control enforcement ordering process. There may be, for example, a determination made that an individual User Representative may only order some AVs and not others. In general, if any such restrictions are to be imposed on the ordering process, they need to be identified and input to the subscriber access control ordering process.

2.1.1.3 Partition IBAC Requirements Definition

In addition to defining the rule-based portion of the SDNS component enforcement it is necessary to identify the identity based portion. The IBAC portion of the policy representation is outside of ACIS. This function defines the nature of the subscriber identification (e.g., a military IP address, an ISO IP address, or X.25 address) and the associated rule-set used to determine if this portion of the SDNS enforcement expression has been satisfied (e.g., address must be found on an access control list before communication is authorized).

2.1.1.4 Access Control Ordering Form Design

Once the partition rule-based and identity-based access control requirements have been specified it is possible to design access control ordering forms which support these requirements. These forms are used by the SDNS ordering functions. They can either be paper or on-line forms.

The ordering forms will include both IBAC and RBAC information. Presumably the RBAC information will be structured in a format similar

to that of the ACIS representations to aid the User Rep in filling out the subscriber order form and to aid in encoding of the access control information in the keying material during the ordering process. These forms will be used during the ordering process (blocks 3 through 5 of Figure 2) to help ensure the correct entity attributes are reflected in the access control information distributed to each entity in the system.

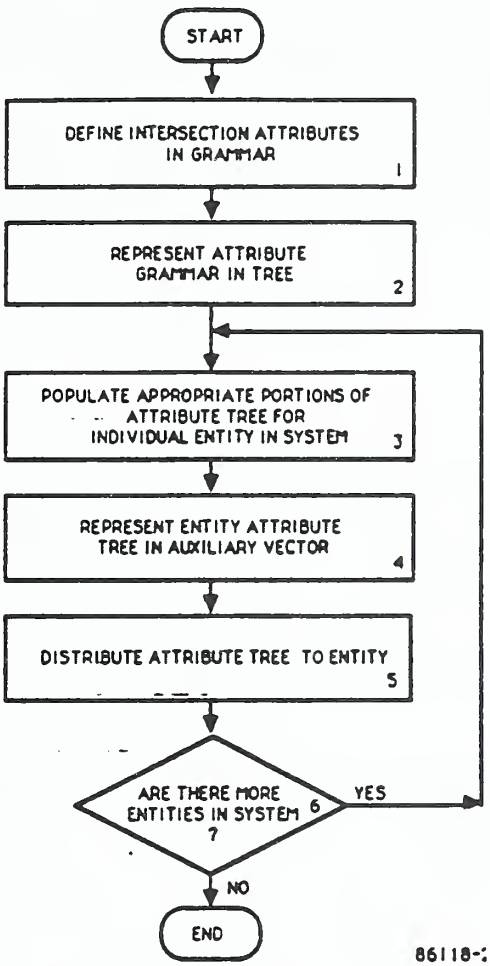


Figure 2 Attribute Definition and Distribution (Process 1)

2.1.2 DEFINE LOCAL ACCESS CONTROL POLICY REQUIREMENTS

This function is analogous to those functions just discussed at the partition level. In this case, enforcement requirements are generated for local access control policies by the local authority just as was done at the partition level. For type 1 equipments, local access control policy is generally a refinement of partition policy. This means that the local

policy serves as an extension of the partition policy. It is also possible for the local policy to be completely disjoint from the partition policy, although this is probably more likely for type 2 devices rather than type 1. This function is not required in those systems which do not support a local access control policy.

2.2 ACIS Grammar Definition

ACIS is derived from general compiler theory principles and is meant to support the rule-based portions of an access control policy. The remaining portion of Section 2, and Sections 3, 4, and 5 assume that the reader is already familiar with the compiler concepts contained in the discussions. If the concepts are unclear, the reader is referred to Section 6 where a brief summary of the compiler principles underlying ACIS is presented.

In this section, the formal definition of the ACIS grammar requirements is presented. These requirements include definition of the characteristics of an ACIS grammar, discussion of the grammar operators, a description of the validation process, and a discussion of design strategies to eliminate ambiguities in a grammar definition.

2.2.1 ACIS GRAMMAR REQUIREMENTS

ACIS uses the concept of parse trees and predictive parsers in order to test PDU labels and to compare access attributes in PAA. However, unlike the theoretical treatment in Section 6, ACIS does not require that parse trees be constructed dynamically to test PDU labels. Instead, a single parse tree representing all acceptable PDU labels for the system is constructed at the time the access control policy representation is generated. This single parse tree is the system attribute tree. Also contained in the attribute tree is access control information used during PAA. Part of the tree generation task is to guarantee that the representation is unambiguous and suitable for use as a predictive parser for label testing.

Appropriate portions of this master tree are selected for each entity in the system during the ordering process. Recall that the master attribute tree has already been shown to be unambiguous so subsets of this tree should also be unambiguous. The access control subsets for two entities are then intersected to determine those labels (and PAA access control attributes) that are shared by both entities. The intersection process (which is documented in Section 4) consists basically of comparing the entity attribute tree and pruning off portions from each tree which are not shared by both trees. The output of this process is an enforcement vector which defines those PDU labels which are shared by both entities. This enforcement vector essentially serves as a master parse tree which defines every allowable string shared by both entities. Testing a label corresponds to creating a leftmost derivation for that string using the production shared by both entities.

For the sake of sample, assume that two entities in the system share the following grammar.

A → B C
B → D E
C → F G
D → 1 | 2
E → 3 | 4
F → 5 | 6

G → 7 | 8

The object is to test whether the string 1368 is allowed by the grammar shared by both entities. For a predictive parser such as ACIS it must be possible to create a leftmost derivation for this string from the productions of the grammar. A leftmost derivation is one in which the leftmost nonterminal in each production is always derived first. This is done by beginning with the start symbol A and expanding the leftmost nonterminal in each step of the derivation process. The resulting derivation is

A → BC → DEC → 1EC → 13C → 13FG → 136G → 1368

At each point in the derivation, the choice of production to be used for derivation is guided by the lookahead character. In this particular example, A can only be expanded as BC. Similarly, B can only be expanded as DE but for the nonterminal D there are two possible choices, 1 or 2. The string itself says the choice must be 1. This same process occurs for each of the characters in the string being tested. ACIS grammars have the characteristic that during the derivation process at each point when alternate choices must be selected, there will be only one choice that matches the lookahead character.

A leftmost derivation also corresponds to the structure of the parse tree constructed for a predictive parser. The root of the tree corresponds to the start symbol in the grammar. The lower nodes are constructed in a preorder fashion (preorder is also called depth-first order). Thus at each node, the leftmost nodes are derived in a depth-first order. Recall that in depth first order, the nodes of the tree are visited in the order described by the following recursive procedure:

1. Visit the nodes. (This node will be expanded by subtrees below it)
2. Traverse the left subtrees, if any, in depth-first order. (Terminals at the leaves mark the end of traversal).
3. Traverse the right subtrees, if any, in depth-first order. (Again, terminals at the leaves mark the end of the traversal).

Consider the EV represented in Figure 2.2.1-1 which represents the grammar presented earlier. This tree guarantees leftmost derivations when the tree is searched in depth-first order as defined above. The nonterminal represented at each node of the tree is labelled in the drawing. For example, the nonterminal A is defined when the root node is visited in depth-first order. Thus as in the production, A is defined by the

node B and node C. B is defined by nodes D and E which are also visited in preorder and so on.

Highlighted on the figure is a leftmost derivation for the string 1368. This derivation occurs when the nodes are visited in a depth-most fashion. Note that this corresponds to the derivation presented earlier. A more detailed explanation of the latest label testing process is presented in Section 5.

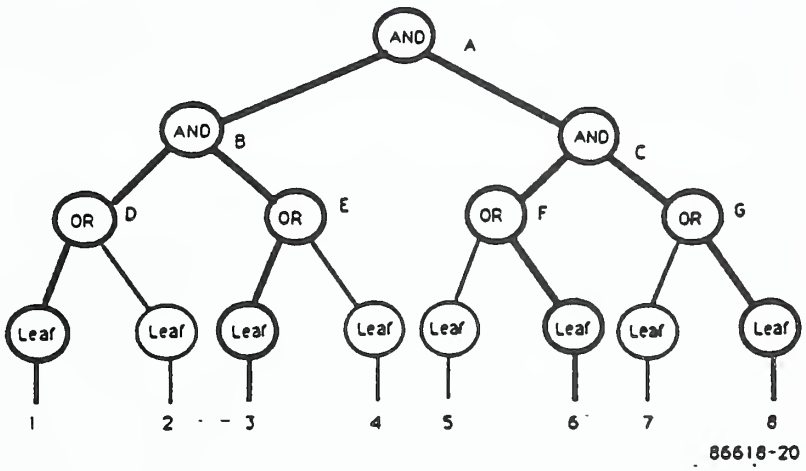


Figure 2.2.1-1 Parse Tree for String 1368

With this background information, the following requirements are imposed upon ACIS grammars:

1. ACIS grammars will be designed to support predictive parsing. Formally they will be a form of LL(1) grammar. The first "L" in LL(1) stands for scanning the string to be parsed from left to right. The second "L" stands for a leftmost derivation. The 1 in LL(1) indicates that a single lookahead symbol may be used to determine the correct production to choose (or equivalently the correct node to select). In other words, it is expressly forbidden for two alternative choices to begin with the same first character.

2. e - productions are not allowed (e stands for the empty string).[1]

[1] This restriction as well as restriction 3 are currently being investigated to see if they can be relaxed somewhat in a future version of

Normally e productions are used during parsing only when all of the non-e productions fail to match then the e production is a legitimate choice. If the e - production was not really intended, then chances are probably good that the rest of the parsing will fail. This is not sufficiently robust when testing a PDU security label. In addition, e productions greatly complicate the design and verification of the grammar definition. A special character, f , is defined in ACIS to indicate the end of a label string. This character can be included in the grammar to explicitly check for the end of a string.

3. Recursive productions are not allowed. Left recursive productions are prohibited because they are not suitable for predictive parsing. Right recursive productions are prohibited because they do not allow parse trees to be statically defined. There are two ways in which to test for recursive productions in general. The first to look for immediately recursive definitions in the grammar productions themselves. Immediately recursive productions are those in which the nonterminal appearing on the left side of a production also appear on the right side of a production. It is still possible for productions to be recursive without appearing to be so. Consider the grammar

   S → aA | b
   A → c | dS

The right recursion is not readily apparent until the definition of S is substituted into the production for A. Doing so yields the grammar

   S → aA | b
   A → c | daA | db

---

this spec to provide additional flexibility. In particular it would be desirable if limited recursion could be provided to allow matching elements of a list until a specific number of elements were found or until the first occurrence of a non-match. A least one match would have to occur though to be acceptable.

In general these kinds of problems can be eliminated if the productions are organized so that the definitions proceed from the start symbol in a linear fashion to the terminal. These kinds of productions are guaranteed to be found during the process of transforming the grammar definition into its ACIS representation. If a recursive production does exist, the leaf corresponding to the recursively defined nonterminal will reference a node that has previously been defined. Graphically the leaf will attempt to point to a node higher in the tree.

4. The right hand side of productions may only contain one type of operator. The production may include more than one instance of a particular operator but only one type may appear. This restriction is imposed so that nesting of operators does not occur in productions and also to aid in the encoding process.

2.2.2 ACE PRODUCTION OPERATORS

Following is a discussion of the operators that are proposed for use in the productions of an ACIS grammar. Note that within an ACIS production only one type of operator may be used. For each operator, a brief description of the operation is provided followed by a description of the syntax used for the operator. Finally an example is presented for each to illustrate its usage.

2.2.2.1 AND Operator

The AND (+) operator is used to denote the concatenation of the symbols appearing on the right side of productions in the order shown. These symbols can be either terminals or nonterminals.

Form:

$symbol_1 + symbol_2 + symbol_3 + \ldots + symbol_n$

Example:

paa_info → accreditor + classification + channel

This production indicates that the nonterminal paa_info is defined by the three nonterminals accreditor, classification, and channel in that order. Each of these nonterminals represents attributes as defined by the production associated with it.

2.2.2.2 OR Operator

This operator is used to denote alternate choices of definitions for the nonterminal appearing on the left side of the production.

Form:

$symbol_1 \mid symbol_2$

Here symbols $symbol_1$ and $symbol_2$ may be either terminals or a nonterminals.

Example:

classification $\rightarrow DE_H \mid AD_H \mid 7A_H \mid 55_H$

The H subscript is used to denote that the value is hexadecimal. This production indicates that the nonterminal classification may be represented by one of the follow values: DE, AD, 7A, or 55.

The AND and the OR operators can be thought of as the fundamental operations that are defined for use in productions. All of the productions supported by ACIS can be defined in terms of these two basic operations. In some cases though, the definition process may be somewhat cumbersome because of the potentially large list of symbols which would be required to be used in productions to represent some relationships. To combat this problem and to provide a slightly higher level of abstraction, the following operators have been defined.

2.2.2.3 Numerical Range

The numerical range operator is used to check that the value in t label string under test falls within the bounds specified by the numerical range operator.

Form:

Num_Range (field width in octets, Upper Limit, Lower Limit)

The field width in octets identifies the length in octets of the numerical value specified in the upper and lower limits and the label string to be tested. The Upper Limit specifies the maximum acceptable value in the label string. The Lower Limit specifies the minimum acceptable numerical value in the label string.

Example:

Num_Range(1, 05H, 0AH)

This expression can be expanded to:

05H | 06H | 07H | 08H | 09H | 0AH

In this example one octet will be removed from the label string and test to ensure that the value falls within the numerical range from 05H through 0AH inclusive.

2.2.2.4 Bit Vector Range

The bit vector range can be used as a more economical representation method of defining set membership than by explicitly

identifying each member of the set. Potentially bit vectors could be used when the number of members that can be identified in a label becomes comparable with the set size. For example, suppose that there are eight members in a set and that potentially all of them could be named within a label. If the members of the set are identified in three bits and each of the members was explicitly named, it would take 8 * 3 bits = 24 bits to list all of the members. However if each member was assigned a particular bit position then naming all eight members takes only 8 bits. The presence of a 1 in a particular bit position reflects the presence of that member in the label while a 0 reflects its absence. The overhead associated with the bit vector is always 8 bits even if the list is empty whereas an explicit list would require zero bits to represent an empty list. The bit vector range representation however, does not allow the same degree of granularity as the list since it is impossible to exclude particular members falling within the range.

This operator can also be used to express a COMPUSEC dominance relationship. The upper value is said to dominate the lower value. The upper limit identifies the maximum number of items that are authorized. A 1 bit identifies authorization for that category. The lower value identifies the minimum number of items that must be identified in a label. So long as the minimum number of items are identified and as long as none are identified that are not authorized the label is acceptable.

Form:

BV_Range (field width in octets, Upper Limit, Lower Limit)

The field width in octets refers to the numbers of octets in each of the terminal strings represented by Upper Limit and Lower Limit. Each of these parameters is assumed to be of the same fixed size represented by field width.

Upper Limit and Lower Limit are meant to represent the terminal strings defining the upper and lower range limits. Upper limit refers to the value which represents the numerically greater value if the terminal string is considered an unsigned integer. Lower limit represents the lower numeric value if Lower Limit is similarly viewed as an unsigned integer.

Example:

BV_Range (1, 72H, 02H)

This expression can be expanded to:

02H | 12H | 22H | 32H | 42H | 52H | 62H | 72H

The concept is illustrated by the Figure 2.2.2.3-1 in which the limits of the range are represented as binary numbers.

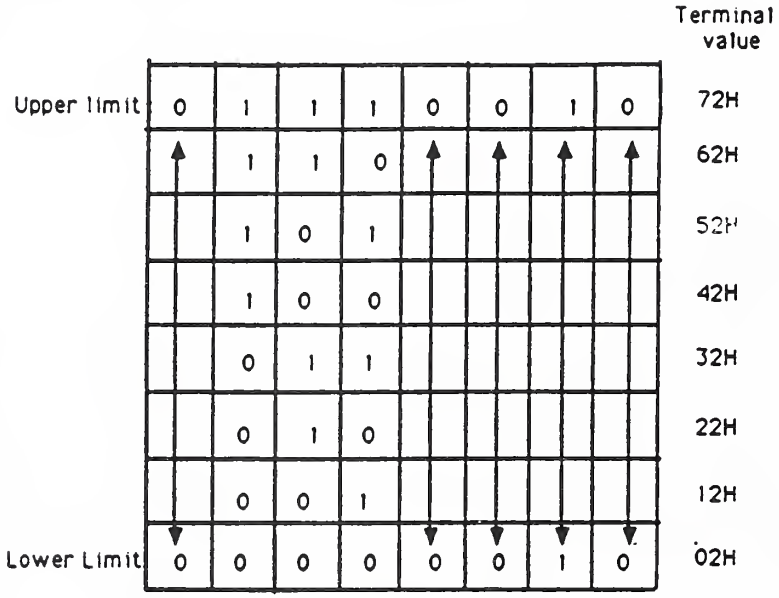


Figure 2.2.2.4-1 Bit Vector Range

The value of the range expression can be derived by performing a bitwise-and of the two values and by setting those bit positions that are a "one" in the upper limit and a "zero" in the lower limit to a don't-care (represented by an x). This means that either value in that bit position is valid. Thus, for the example cited, the resulting range of acceptable values when expressed in binary form is (0XXX0010). This range is explicitly listed in the expanded production.

2.2.2.5 N_OF

This operation is used to specify an N-long list of members from some set of members. Thus, for example, the 1_OF operator is used to specify one member out of an list of possibilities. The 2_OF operator is used to specify two members from a set of members and so on for the other possible N_OF operators.

Form: N_OF (member count, count representation, (set definition))

The member count identifies how many values from the set definition will follow the count in the member list.

Set definition has the form:

$set\_member_1 \mid set\_member_2 \mid \ldots \mid set\_member_n$

The symbol set_member is used as placeholder for the actual token string representation of the set member. An equivalent production may be used for the set definition, i.e., if the production set $\rightarrow \varepsilon \mid set\_member_1 \mid set\_member_2 \mid \ldots \mid set\_member_n$ is used the nonterminal, set, may be used where set definition appears in the format statement. Note that set members in the above definition must be terminals.

This operator is used to represent strings of the following form:

N choices_subset

N indicates the number of members (e.g., 1, 2, 3, etc.) to follow as specified by the N_OF operator and choices_subset is the list of possible strings that can be derived by taking the set definition and choosing N at a time. The ACIS machine operating in the SDNS component does the following when it encounters this operator. It first checks that number of members to follow matches the number specified by the N_OF operator (and represented by the literal defined as count representation). If the number matches, then the N members are each tested for membership within the specified set. Lastly the members are tested to ensure that they are not duplicates of one another.

Example:

N_OF (2, 02H, (DEH | ADH | 7AH | 55H))

This says that strings containing an indication of 2 members followed by 2 members drawn from the set two at a time are acceptable. This means that any of the following strings are acceptable.

2 DEADH | 2 DE7AH | 2 DE55H | 2 ADDEH | 2 AD7AH | 2 AD55H | 2 7ADEH | 2 7A55H | 2 55DEH | 2 55ADH | 2 557AH

2.2.2.6 Don't Care

This operator is used to enable the correct parsing of a label while allowing portions of it to be untested. This facility is used if the label contains fields that are informational and not included in the access control checks. These informational fields are identified as don't care fields so that parsing may continue for fields later in the string which may be included in the access control checks.

Form:

DONT_CARE (length in octets)

The DONT_CARE operator is used to allow a match to occur for any value found in the label at that position in the string. Each octet of the field is allowed to range over the values 00H to FFH.

Example:
DONT_CARE (2)

This says any of the string 0000H, 0001H, through FFFFH will be accepted.

2.2.2.7 NOT

The NOT operator is used as a convenient way of excluding a small set of members from a large set. Consider the case in which all members are acceptable except for 1 or 2. These members are identified as excluded members from a more general set definition which includes the excluded members. This operator essentially serves as a macro instruction. The compiler will take the general set definition, remove those members from the excluded set, and turn it into a set definition using the OR operator. Note that the members must be terminals.

Form:

NOT ((exclude_set), (general set))

The exclude_set and the general set may either be nonterminals with their own productions or be a list of member terminals separated by | symbols.

Example:

NOT ((01H, 05H), (set_a))

set_a → 00H | 01H | 02H | 03H | 04H | 05H | 06H

The compiler will turn these two productions into an equivalent one of the form.

set_a' → 00H | 02H | 03H | 04H | 06H

2.2.3 GRAMMAR VALIDATION

Before an ACIS grammar can be deemed acceptable several checks must be performed. They are:

1. The productions must conform to the syntax defined for the operators in Section 2.2.2.
2. Each nonterminal appearing on the right hand side appears on the left side of a production once and only once.
3. The grammar must be shown be unambiguous. This process is relatively straight forward once the grammar is represented as an ACIS tree. Consider the following grammar:

   A → B | C
   B → D + E
   D → X | Y
   E → RS | XTZ
   C → DONT_CARE(1) + Q

The grammar is represented as a tree in Figure 2.2.3-1. The nodes have been numbered to ease the discussion. The process basically consists of identifying the first character beginning each string and seeing if two alternate choices begin with the same first character. The process begins at the leafs. At node 8, for example, the first character that could be generated at this node is x. This set of characters is denoted by (x) shown beside the node. At node 10, the first character of a possible string begins with R. At node 6, any value is acceptable because the node is a don't care operator. Travelling up the tree there are two cases to consider. If the operator is an OR, the first character set becomes the union of the nodes immediately below it. Thus at node 4, the set is composed of X and Y. If the operator is an AND, the first set is becomes the first set of the leftmost subnode since subnodes to the right of this node cannot begin the string. Thus at node 2 the set is composed of (X, Y). Preceding up to node 1, we see there is an ambiguity since the first characters from node 2 are X any Y and these same characters are included from the don't care values from node 3.

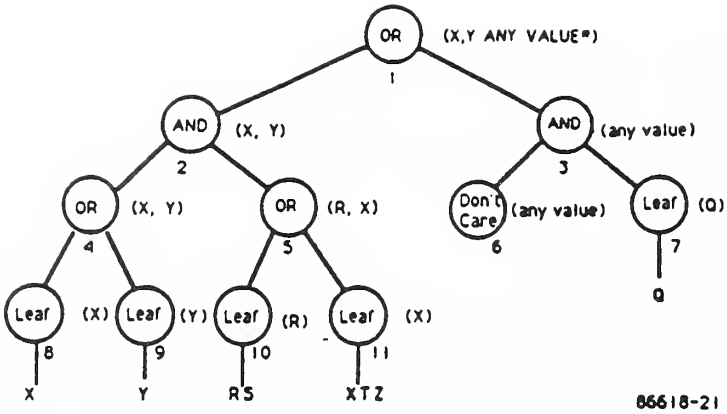


Figure 2.2.3-1 Ambiguity Testing Example

4. We must check that there are no recursive productions. These will be spotted as node trying to point to themselves or to a higher node in the tree (i.e., cycles will appear).

2.2.4 GRAMMAR DESIGN STRATEGIES

After an ACIS grammar has been defined it must be tested according to the requirements in Section 2.2.3. It may occur that during the validation process the grammar is shown to have errors that cause the grammar to be unsuitable for use with a predictive parser without corrections. The following paragraphs describe strategies that can be used to correct some commonly occurring problems.

2.2.4.1 Left Factoring

Left factoring can be used when there are alternate productions for a nonterminal and it is unclear which to use. Left factoring may make it possible to defer the decision of which production to use until the lookahead symbol can determine the choice. Consider a production of the form $A \rightarrow ab_1 \mid ab_2$, it is unclear which alternative to use because both begin with the nonterminal a. This ambiguity makes the grammar unsuitable for predictive parsing. However if the production is rewritten as:

$A \rightarrow ab'$
$b' \rightarrow b_1 \mid b_2$

The decision can be deferred until b' must be expanded. However at this point the beginning characters which can be derived from b_1 and b_2 should determine the appropriate choice.

2.2.4.2 Addition of Special Characters to Resolve Ambiguities

Consider the following portion of an ACIS grammar which was presented earlier in Section 2.2.3.

A → B | C
B → D + E
D → X | Y
E → RS | XTZ
C → DONT_CARE(1) + Q

As the productions currently stand they are unacceptable because the strings derived from B begin with either X or Y and the strings derived from C may begin with any value including X or Y. This means that the production associated with A is ambiguous. To resolve the ambiguity, a terminal other than X or Y can be added before the DONT_CARE field in the

production for C. For example, if the production is changed to C → F + DONT_CARE(1) + Q, the ambiguity is resolved. Strings using the nonterminal B begin with X or Y while the the strings derived from nonterminal C begin with F.

2.3 ACIS Grammar Examples

Presented are two preliminary examples of access control information that may be represented using ACIS. The first is an example of access control attributes pertinent to the PAA process. The second is an example of PDU labelling information that would be used in PAE. The labelling information can be added as subtrees to the PAA information as an extension to it. This provides a complete representation of the RBAC policy. The goal is to accomplish that in the next version of this spec.

2.3.1 PAA EXAMPLE

Figure 2.3.1-1 presents an example of the relationships access control attributes that could be used in the PAA process for a community partition. Shown on the figure is a mapping of the access control attributes to the four tiered model of the Access Control Concept Document. The tier 1 and 2 information would be contained in the Primary Vector Supplied by the KMS while the tier 3 information would be contained in the auxiliary vector generated by the Local Authority. The auxiliary vector would also include the tier 1 and 2 information since the tier 3 information is an extension of those tiers.

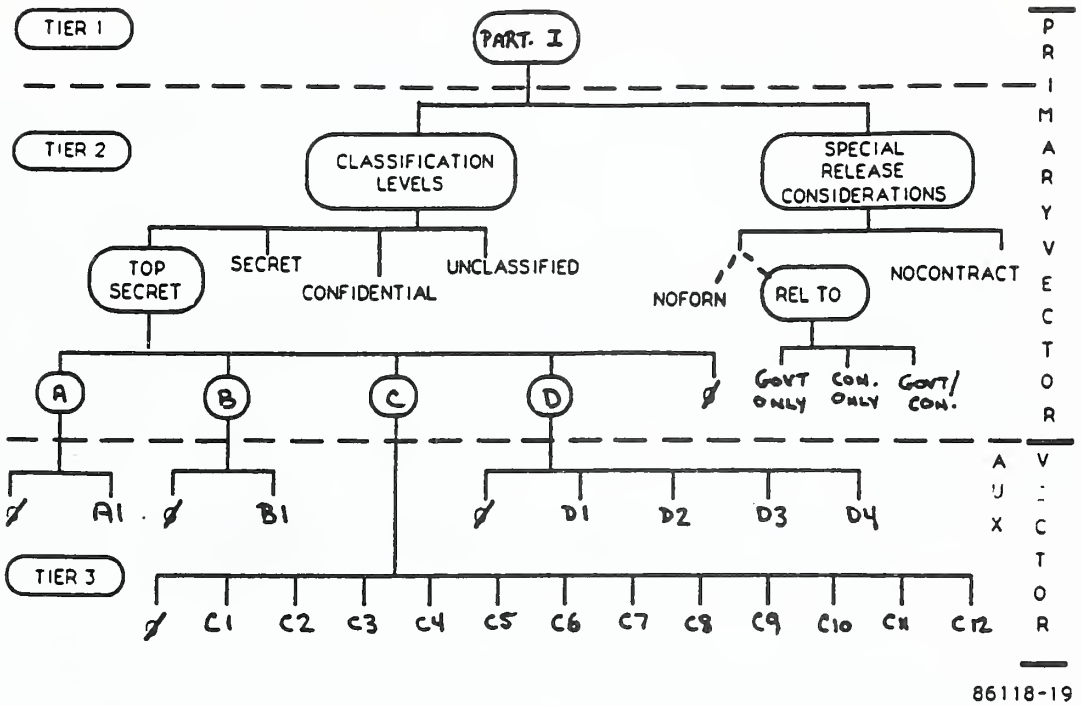


Figure 2.3.1-1 Example of PAA Information for a Community Partition; Breakdown of Tiers 1 through 3

2.3.2 PAE EXAMPLE

Figure 2.3.2-2 shows a sample policy representation of PDU labelling information. The grammar of Figure 2.3.2-1 is represented as a tree in Figure 2.3.2-2. The nodes have been marked to indicate their corresponding productions in the grammar. The policy representation is for illustration purposes. An actual representation may be considerably larger than the one shown.

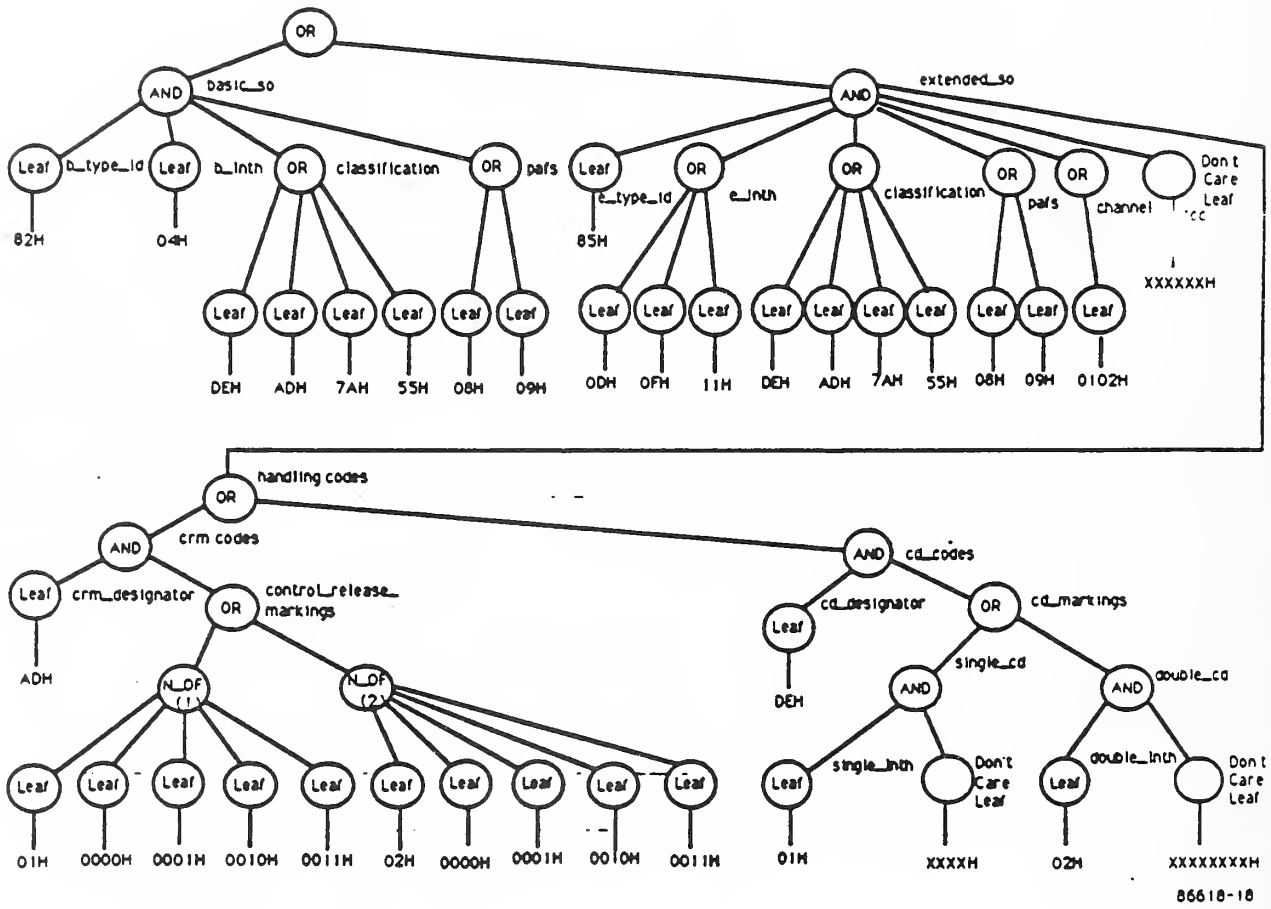


Figure 2.3.2-2 PAE Example Shown In Tree Format

The tree graphically represents the acceptable PDU labels per the policy representation. There are two acceptable labels defined for this system. The first is the basic option which is indicated by a 82H appearing as the first character of the label. The second is the extended security option which is indicated by a starting character of 85H. Notice that the labels themselves appear at the leaves of the tree. For example, the basic security options is made up of the string 82H followed by 04H followed by (DEH or ADH or 7AH or 55H) followed by (08H or 09H). This defines the total set of acceptable labels that begin with an 82H.

Figure 2.3.2-1 Example ACIS PAE Grammar Definition

```
pae_info  Æ  basic_so | extended_so

basic_so Æ b_type_id•b_lnth•classification•pafs
b_type_id  Æ  82H
b_lnth  Æ  04H
classification  Æ  DEH | ADH | 7AH | 55H
extended_so  Æ  e_type_id•e_lnth•classification•pafs•channel•tcc•handling_codes

e_type_id  Æ  85H
e_lnth  Æ  ODH | OFH | 11H
pafs  Æ  08H | 09H
channel Æ 0102H
tcc  Æ  DONT_CARE(3)
handling_codes  Æ  crm_codes | cd_codes
crm_codes  Æ  crm_designator•control_release_markings
crm_designator  Æ  ADH
control_release_markings  Æ  crm_single_marking | crm_double_marking
crm_single_marking  Æ  N_OF(1, 01H, (0000H | 0001H | 0010H | 0011H)
crm_double_marking  Æ  N_OF(2, 02H, (0000H | 0001H | 0010H | 0011H)
cd_codes Æ cd_designator • cd_markings
cd_designator Æ DEH
cd_markings Æ single_cd | double_cd
single_cd Æ single_lnth • DONT_CARE (2)
double_cd Æ double_lnth • DONT_CARE (4)
single_lnth Æ 01H
double_lnth Æ 02H
```

3.0 ENCODING OF ENTITY ATTRIBUTES

3.1 Order Subscriber Access Control Attribute Tree

Within the SDNS Access Control Subsystem, the partition and local access control policies are supported by different ordering functions. They are: ordering of subscriber partition access control attributes and ordering of subscriber local access control attributes. Each of these is discussed in greater detail in the following paragraphs.

3.1.1 ORDERING OF SUBSCRIBER PARTITION ACCESS CONTROL PERMISSIONS

This function can be decomposed into three activities: generating a subscriber partition access control order, encoding the subscriber access control order, and generating the primart certificate.

Partition Ordering Forms, Local Domain Authority Identifier, Partition Ordering Restrictions, and the subscriber (entity) partition access control attributes are used to generate a partition access control order for a particular subscriber entity. The Partition Order Forms provide a template into which are placed the various access control attributes relevant to a particular subscriber entity out of the total set of possible attributes which were defined for the partition. Essentially this process consists of customizing the access control attributes that exist within a partition to a particular subscriber entity. The Partition Order Restrictions (if there are any) determine whether there are any restrictions upon what access control attributes a User Representative (UR) is allowed to place in the Partition Ordering Forms. The Local Domain Authority ID is optional information which is required to support the local access control system. The Local Domain Authority ID identifies the authorized Local Authority which is able to generate Auxiliary Vectors (AVs) containing local access control information. Essentially the information in an auxiliary vector can be believed to be authentic because the agent that generated the AV (i.e., the Local Domain Authority) was identified in the primary certificate. Associated with the Local Authority is a Message Signature Key. This key is used to generate a digital signature which ensures the integrity of the auxiliary vector.

The final phase of the partition access control ordering process is the inclusion of the encoded subscriber access control order in a primary certificate. This function is outside of the purview of SDNS. It is performed by the Key Management System.

It is unclear whether ACIS will be used to represent the access control attributes in the Primary Certificate. Currently the attributes are defined as a fixed series of fields that are included in the certificate. It is possible for this information to be duplicated in the AV as part of the local policy representation as was mentioned earlier.

3.1.2 ORDERING SUBSCRIBER LOCAL ACCESS CONTROL ATTRIBUTES

This function is very similar to the Partition Ordering process described in section 3.1.1. The Local Access Control Policy ordering process is performed under the auspices of the local authority instead.of the partition authority.

The process of generating the subscriber local access control order and the order encoding process are essentially the same as were done for partition ordering except that the access control attributes support local policy instead of partition policy.

The AV is used mostly as a vehicle for distributing local access control policy information to the SDNS component. This includes both rule-based and identity-based access control information. Because this process is under a local authority's control, greater variation is expected in the policy expression. In addition, it is possible for the AV to contain additional information since this is strictly a local concern. It is assumed, however, that the rule-based local access control information will be represented in an ACIS format to ensure interoperability at local rule-based policy level.

3.2 Entity Attribute Encoding Rules

This section details the algorithms for encoding entity attribute trees in an AV and for recovering the entity attribute tree from its encoded form in the AV. Before the algorithms are presented a brief discussion of the approach and the encoding rules are presented.

3.2.1 APPROACH

The following encoding rules have been designed to resemble the NBS.1 ASN.1 methodology. This method consists of preceding all data fields with type and length fields. This approach is easily expandable and places no restrictions on the format of the actual data. The type field identifies the type of data within the data field, such as integer or octetstring. The length field specifies the length in bytes of the data and the data field contains the actual data. The standard ASN.1 definition specifies the type and length fields to be at least one byte each.

Since the number of bytes used to encode the entity attribute trees is critical, the ASN.1 approach has been modified to reduce the number of bytes necessary for the type and length fields. This is achieved by redefining the type identifiers and lengths of the type and length fields. The data types are all defined by one nibble and wherever possible the data length field is encoded in the following nibble (if the length of data is < 16 bytes). This reduces by half the number of bytes necessary to encode the type and length fields in the AV over the standard ASN.1 encoding.

3.2.2 ENCODING RULES

A minimum of three categories of type fields are necessary to encode an entity attribute tree. The fourth type has been added to allow encoding of Leaf data with a length greater than 15. These include a control data field, a leaf data field and a don't care data field. The hex definitions of the type fields are shown below:

Type Field Definitions

D = Control Data type
E = Leaf Data type
F = Don't Care Data
0 = Reserved (See discussion for Leaf Data Type)

The control data field is used to encode control characters which represent nodes other than leaf nodes. These leaf nodes may contain data (in which case they would be encoded using an E or 0 type field indicator) or may not (in which case they would be encoded using an F type field indicator). Examples of data encoded using a Control Data type field are an "OR" or an "AND" node. The control characters are defined to be one nibble long and where possible multiple control characters are placed together within a single data field. If there are an odd number of of control characters then a null character is used to pad the remainder of the byte within the data field.

Control Data fields are preceded by a byte containing a Control Data type indicator (D hex) in the upper nibble and a indication of length of the control data (in bytes) in the low order nibble. This allows a maximum of 15 bytes or 30 control characters to be represented in the datafield. If a greater number of control nodes are to be represented, a second control data field representation can be concatenated to the previous control field encoding. The definitions of the control characters used in the control data field are shown below. The values 6 through E hex can be used to represent future operators that may defined for ACIS. The value F does not actually correspond to any actual node within a tree but is used rather for control information during the process of generating a tree from its encoded representation. Its use will be illustrated later.

Control Character Definitions
(D, E and F may have different definitions as control characters.)

0 = null
1 = OR node
2 = AND node
3 = NUMERIC RANGE node
4 = BIT VECTOR RANGE node
5 = N-OF node

.
.
.

F = backup (return) to parent node

The leaf data field contains data to be placed into a leaf node. A special control character has not been defined to indicate a "Leaf" node. It is not necessary to indicate a "Leaf" node since whenever a leaf data field is encountered it is necessary to create a LEAF node to store the leaf data. As with the control data field, the data field is preceded by a byte containing the leaf data type in the high order nibble and the length of the leaf data in the low order nibble. However, if the leaf data is longer that 15 bytes then the leaf data type is placed in the low order nibble with the high order nibble set equal to ZERO. The length of the leaf data is then encoded within the next byte. This encoding assumes leaf data will never be greater than 255 bytes.

The don't care data field is similar to a leaf data field. Whenever a don't care data field is encountered a don't care LEAF_node is created and the data (the number of don't care bytes) is stored to the node.

A number of operators have control information associated with them. For example the two range operators (the numeric and bit vector range) have an indication of the upper and lower range values. This control information will be encoded in leaf nodes associated with the particular operator. Thus, for the examples just cited, each of the range operator nodes would have two leaf nodes below the range node to indicate the top and bottom of the range. There is also control information associated with the N_OF operator.

3.2.3 ACIS TREE TO ENCODED STRING

Figures 3.2.3 - 1 though 3.2.3 - 4 present a flow chart of the algorithm to transform a n ACIS tree to an encoded string. The titles for Figures 3.2.3 - 2 through 3.2.3 - 4 are drawn from the names by which they

are referenced in the flowcharts.

The control buffer and the AV buffer are global structures which need to be initialized before the start of the transformation. The control buffer is used for the temporary storage of control characters before they are encoded into a control data field. The AV buffer is used to build the AV string a segment at a time. A segment consists of type, length and data fields. When the transformation is complete the AV buffer contains the completely encoded AV string.

The encode tree function is initially invoked with the root node, and then calls itself recursively for each subnode until all subnodes have been processed. Each and every subnode of a parent node is processed to the lowest possible child node (a LEAF) before returning to the parent. The control characters necessary to descend down the tree are saved in the control buffer until a LEAF node is reached. When a LEAF node is reached the control characters in the control buffer are encoded into a control data field (with preceding type and length fields) and are added to the AV buffer. The control buffer is then cleared. Next, the LEAF data is encoded into a LEAF data field and added to the AV buffer. After the LEAF data is added to the AV then the encode tree function returns to the parent of the LEAF node and checks for more subnodes. Each time the recursive encode tree function ends, it returns to the parent node (immediately above the node being processed) until it finally reaches the root node and is complete.

Note: Multiple LEAF nodes below a single parent will not have control data to encode between the LEAF data fields.

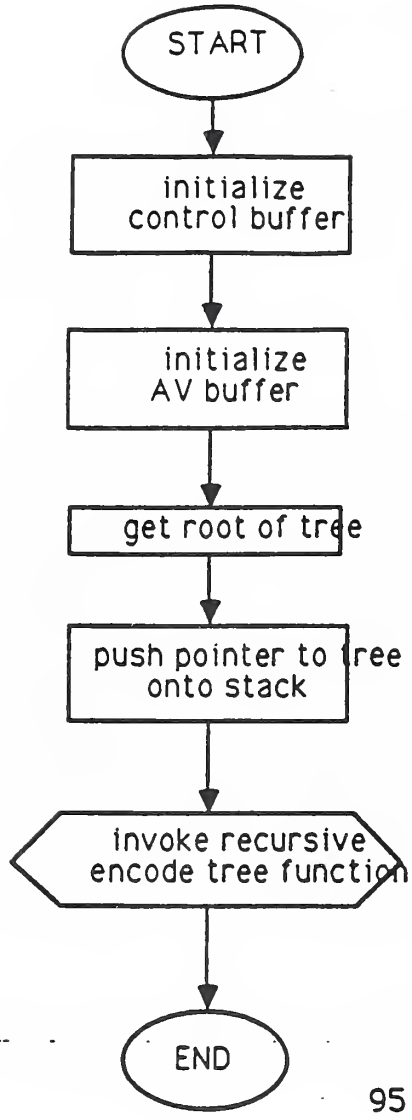


Figure 3.2.3 - 1 ACIS Tree to Encoded String

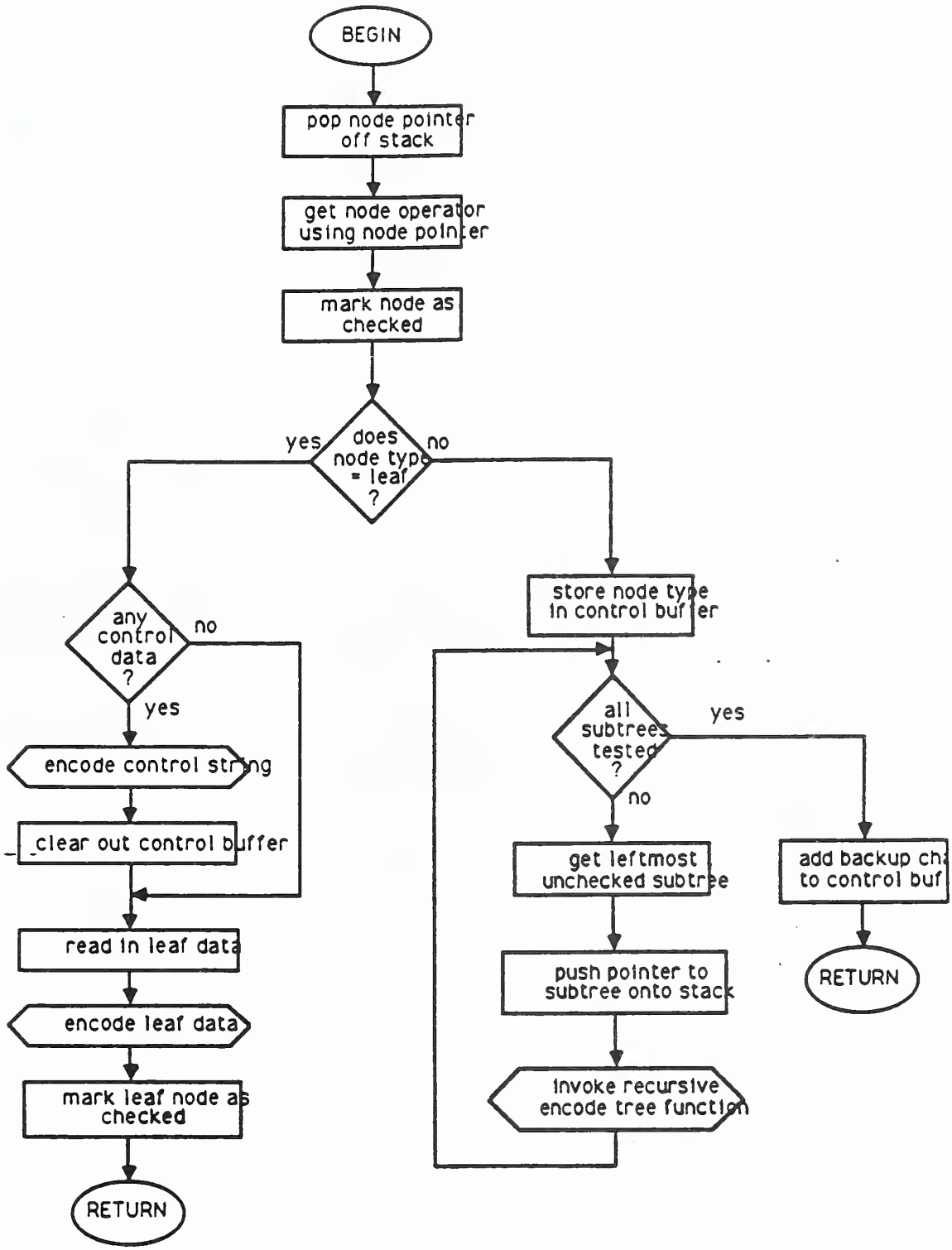
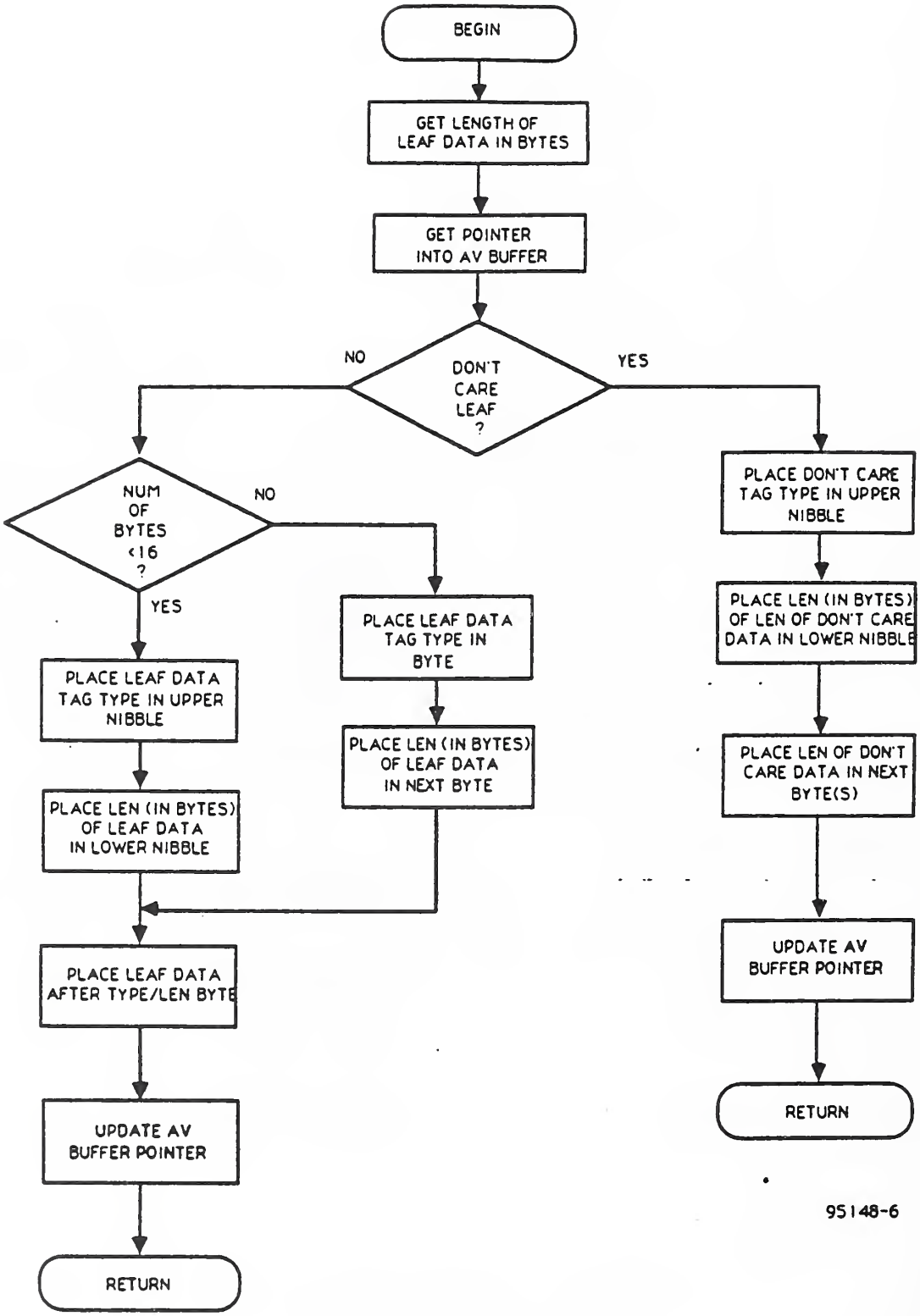
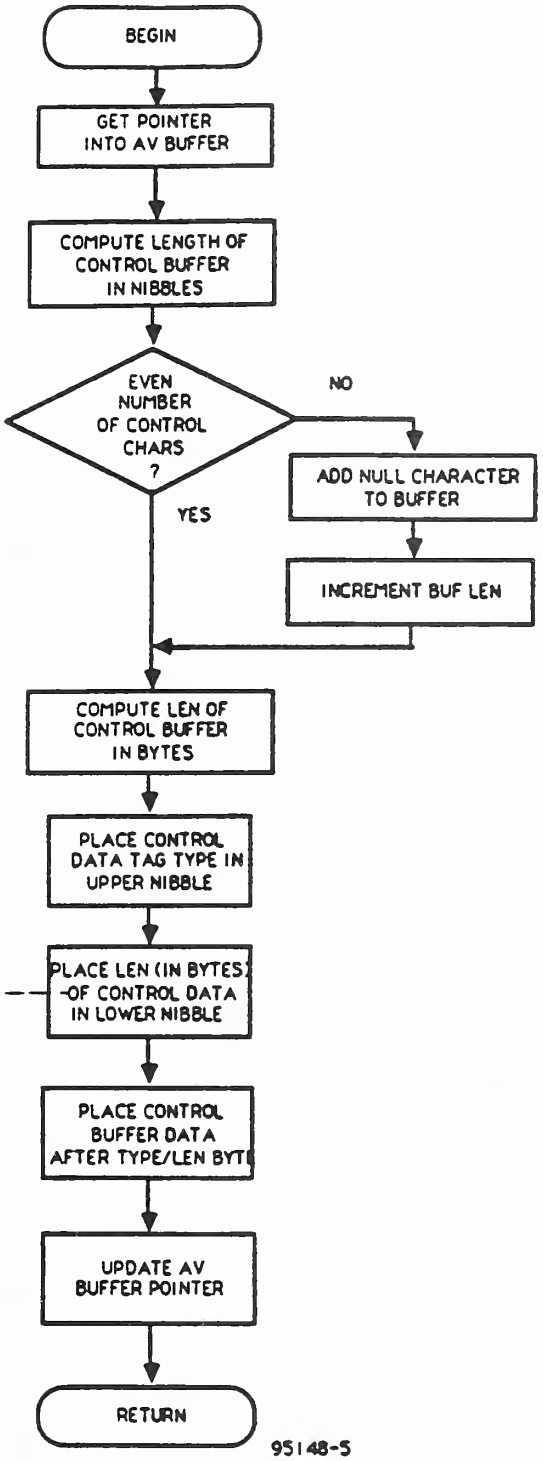


Figure 3.2.3 -2 Recursive Encode Tree Function



3.2.3 - 3 Encode Leaf Data



3.2.3 - 4 Encode Control Data

3.2.4 ENCODED STRING TO ACIS TREE

Figure 3.2.4 - 1 presents a flowchart of the algorithm to transform an encoded string into a Linear tree.

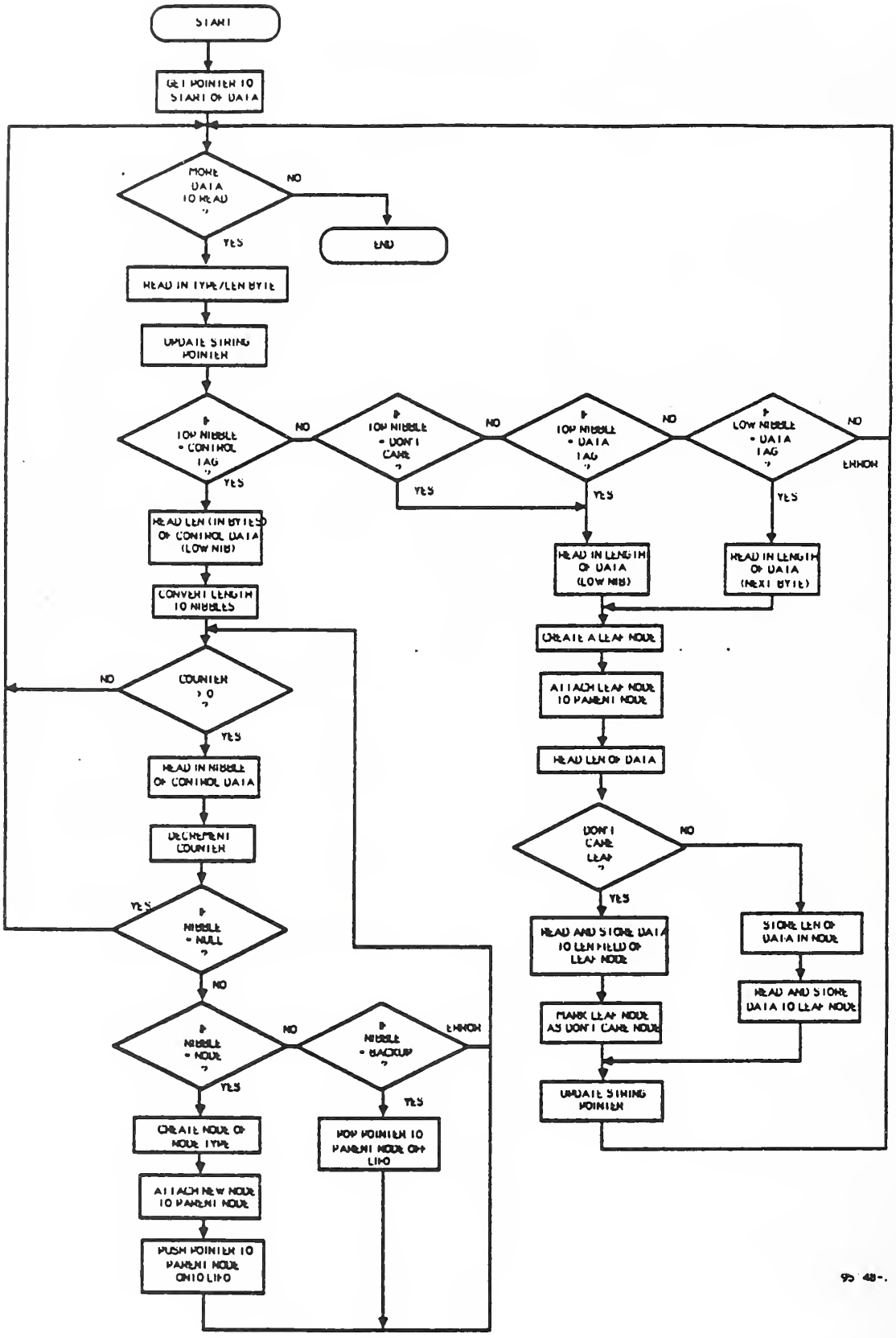
The AV string contains consecutive segments of control and leaf data. The string is processed a segment at a time until the end of the string is reached. Each segment contains a type, length and data field. The type and length fields are descriptors of the data field.

The first field of the segment is the type field and it identifies the type of data within the data field. The second field of the segment is the length field which specifies the number of bytes of data within the data field. When the length of the data is less than 16 bytes the type and length fields are placed in the same byte. Otherwise, the type and length fields are placed in consecutive bytes.

The type field also dictates the utilization of the data within the data field. The control data type specifies that the data field is to be processed a nibble at a time with each nibble representing a control character (as defined above). Each time a control character specifies the creation of a node, the node is initialized and attached to the previously existing node. The new node is now considered the parent node and the next control character is associated with the new node. A Last In First Out (LIFO) queue is used to save the node pointers while descending down the tree. Accordingly, each time a control character specifies the creation of a node, a pointer to the parent node is pushed onto the LIFO. The tree is ascended by popping a pointer to the parent node off of the LIFO. This is implemented by defining a backup control character. A backup control character specifies to pop a node pointer off of the LIFO queue. Subsequent control characters are then associated with the popped node.

The changing of levels in a tree only occurs when processing control data. Leaf and Don't Care fields simply specify to create a LEAF node, store the leaf data to the node and attach the LEAF node to the present node. The level in the tree does not change. This allows multiple LEAF nodes to be placed below a single parent without additional control data. The level in a tree is only changed when "F" in a control data field is encountered. This control character is used to indicate that the algorithm should "back up" to the parent of the current node.

Whenever control or leaf data is processed to completion the pointer to the AV is adjusted to point past the end of the data field to the next segment in the AV string.


START
GET POINTER TO START OF DATA
MORE DATA TO READ ?
NO
YES
END
READ IN TYPE/LEN BYTE
UPDATE STRING POINTER
IF TOP NIBBLE = CONTROL TAG ?
NO
YES
IF TOP NIBBLE = DON'T CARE ?
NO
YES
IF TOP NIBBLE = DATA TAG ?
NO
YES
IF LOW NIBBLE = DATA TAG ?
NO
YES
ERROR
READ LEN (IN BYTES) OF CONTROL DATA (LOW NIB)
CONVERT LENGTH TO NIBBLES
COUNTER > 0 ?
NO
YES
READ IN NIBBLE OF CONTROL DATA
DECREMENT COUNTER
IF NIBBLE = NULL ?
YES
NO
IF NIBBLE = NODE ?
NO
YES
IF NIBBLE = BACKUP ?
ERROR
YES
CREATE NODE OF NODE TYPE
ATTACH NEW NODE TO PARENT NODE
PUSH POINTER TO PARENT NODE ONTO LIFO
POP POINTER TO PARENT NODE OFF LIFO
READ IN LENGTH OF DATA (LOW NIB)
READ IN LENGTH OF DATA (NEXT BYTE)
CREATE A LEAF NODE
ATTACH LEAF NODE TO PARENT NODE
READ LEN OF DATA
DON'T CARE LEAF ?
NO
YES
READ AND STORE DATA TO LEN FIELD OF LEAF NODE
MARK LEAF NODE AS DON'T CARE NODE
STORE LEN OF DATA IN NODE
READ AND STORE DATA TO LEAF NODE
UPDATE STRING POINTER

3.2.5 EXAMPLE TRANSFORMATION OF ENCODED STRING TO ACIS TREE

The following example is modeled from the leftmost branch of figure 2.3.2-2. All values are represented in hexidecimal format.

Key:

T/L = Type/Length
Fn = Field Number n

```
| D1 | 12 | E1 | 82 | E1 | 04 | D1 | 10 |
  T/L   F1   T/L   F2   T/L   F3   T/L   F4

| E1 | DE | E1 | AD | E1 | 7A | E1 | 55 |
  T/L   F5   T/L   F6   T/L   F7   T/L   F8

| D1 | F1 | E1 | 08 | E1 | 09 |
  T/L   F9   T/L   F10  T/L   F11
```

= 22 bytes

1st field = control data of length 1 (byte)
1st nibble = 1 = build an OR node
2nd nibble = 2 = build an AND node (below the OR node)

2nd field = leaf data of length 1 (byte)
build a LEAF node (below the AND node)
read in data (82H) and attach to LEAF node
return (to AND node)

3rd field = leaf data of length 1 (byte)
build a LEAF node (below the AND node)
read in data (04H) and attach to LEAF node
return (to AND node)

4th field = control data of length 1 (byte)
1st nibble = 1 = build an OR node (below the AND node)
2nd nibble = 0 = null (padding)

5th field = leaf data of length 1 (byte)
build a LEAF node (below the OR node)
read in data (DEH) and attach to LEAF node
return (to OR node)

6th field = leaf data of length 1 (byte)
build a LEAF node (below the OR node)
read in data (ADH) and attach to LEAF node
return (to OR node)

7th field = leaf data of length 1 (byte)
build a LEAF node (below the OR node)
read in data (7AH) and attach to LEAF node
return (to OR node)

8th field = leaf data of length 1 (byte)
build a LEAF node (below the OR node)
read in data (55H) and attach to LEAF node
return (to OR node)

9th field = control data of length 1 (byte)
1st nibble = F = backup to next highest node (AND node)
2nd nibble = 1 = build an OR node (below the AND node)

10th field = leaf data of length 1 (byte)
build a LEAF node (below the OR node)
read in data (08H) and attach to LEAF node
return (to OR node)

11th field = leaf data of length 1 (byte)
build a LEAF node (below the OR node)
read in data (09H) and attach to LEAF node

3.2.6 EXAMPLE TRANSFORMATION FROM ACIS TREE TO ENCODED STRING

The following example is again modeled from the leftmost branch of figure 2.3.2-2.

process	command buffer
get top node of tree	
save top node type in command buffer	1
go to leftmost unchecked subnode	
save node type in command buffer	12
go to leftmost subnode	
subnode = LEAF, so encode command buffer	
encoded data = D1 12	
clear command buffer	NULL

read and encode leaf data
mark leaf node as checked

encoded data = E1 82

```
backup a node
go to leftmost unchecked subnode
subnode = LEAF and command buffer is empty
read and encode leaf data
mark leaf node as checked

encoded data = E1  04

backup a node
go to leftmost unchecked subnode
save node type in command buffer                1
go to leftmost subnode
subnode = LEAF, so encode command buffer

encoded data = D1  10

clear command buffer                            NULL
read and encode leaf data
mark leaf node as checked

encoded data = E1  DE

backup a node
go to leftmost unchecked subnode
subnode = LEAF and command buffer is empty
read and encode leaf data
mark leaf node as checked

encoded data = E1  AD

backup a node
go to leftmost unchecked subnode
subnode = LEAF and command buffer is empty
read and encode leaf data
mark leaf node as checked

encoded data = E1  7A

backup a node
go to leftmost unchecked subnode
subnode = LEAF and command buffer is empty
read and encode leaf data
```

mark leaf node as checked

encoded data = E1 55

```
backup a node
node has no more unchecked subnodes so,
mark node as checked, backup a node          F
go to leftmost unchecked subnode
save node type in command buffer             F1
go to leftmost subnode
subnode = LEAF, so encode command buffer

encoded data = D1  F1

clear command buffer                         NULL
read and encode leaf data
mark leaf node as checked

encoded data = E1  08

backup a node
go to leftmost unchecked subnode
subnode = LEAF and command buffer is empty so,
read and encode leaf data
mark leaf node as checked

encoded data = E1  09

backup a node, save in command buffer
node has no more unchecked subnodes so,          F
mark node as checked, backup a node          FF
at top node so done
(no need to encode rest of control characters)
```

3.2.7 POSSIBLE EXPANSIONS OF ENCODING RULES

1. Need a length of more than 16 sequential control characters.

Simply add another field of control data immediately following the first.

Example coding:

```
| DF | 12 34 56 78 9A BC DE F0 | DF | 12 34 56 78 9A BC DE F0 | ...
  T/L              data              T/L              data
```

2. Need more that 16 possible control characters.

Simply add another tag type of data and define new control characters (without changing the original definitions).

C = Control Data (in bytes) tag type
D = Control Data (in nibbles) tag type
E = Leaf Data tag type
F = Don't Care Leaf Data

control character definitions
(Here C, D and E may have different definitions as control characters.)

00 = null
01 = Build OR node
02 = Build AND node
03 = Build a NUMERIC RANGE node
04 = Build a BIT VECTOR RANGE node
05 = Build a N-OF node

.
.
.

0F = backup to parent node

.
.
.

FF = ??

Example coding:

| C2 | 01 02 | E1 | 82 | E1 | 04 | C1 | 01 |
T/L data T/L data T/L data T/L data

| E1 | DE | E1 | AD | E1 | 7A | E1 | 55 |
T/L data T/L data T/L data T/L data

| C2 | 0F 01 | E1 | 08 | E1 | 09 |
T/L data T/L data T/L data

= 24 bytes

Although this example shows every control character represented by a byte, the nibble representation could be used in fields that do not contain byte long control characters.

4.0 COMPARING OF ACCESS CONTROL ATTRIBUTES

Figure 4-1 presents a flow chart of the ACIS portion of PAA. The first three boxes on the left side of the figure apply to the communication initiator while the right three boxes show the actions taken by the communication request recipient.

When the initiator desires to establish communication it first obtains its own local entity attribute tree (which is contained in the entity's auxiliary vector) and sends it to the remote entity with whom communication is desired. When the remote entity receives the initiator's attribute tree (Receive Remote Entity Attribute Tree) it gets a copy of its own attribute tree and sends it to the initiator.

Associated with receiving a remote entity's attribute tree is processing required to ensure the integrity and authenticity of the AV. This process includes verifying the signature used on the AV and checking that the Local Domain Authority is an authorized authority. In addition to that processing is processing which is necessary to extract the RBAC and IBAC relevant information from the auxiliary vector. Since the attribute tree has to be encoded in some fashion in order to place it into the auxiliary vector, it is necessary to undo the encoding process to recover the attribute tree. The requirements for extracting the RBAC and IBAC

information and those associated with undoing the encoding process to recover the attribute tree still need to be supplied.

Once the remote and local attribute trees are obtained the Intersect Local and Remote Entity Attribute Trees function depicted in Figure 4-2 is invoked. A copy of the local attribute tree is passed to this function. This copy of the local attribute tree serves as the basis of comparison with the remote entity attribute tree. If the intersection is non-null, it will also become the PDU labelling testing portion of the EV. The intersection function returns a indication of whether the intersection process yields a non-null intersection. If a non-null result is obtained then communication is authorized and the resulting EV is stored for later use when testing PDU labels. If a null intersection results then it is necessary to perform the necessary error handling.

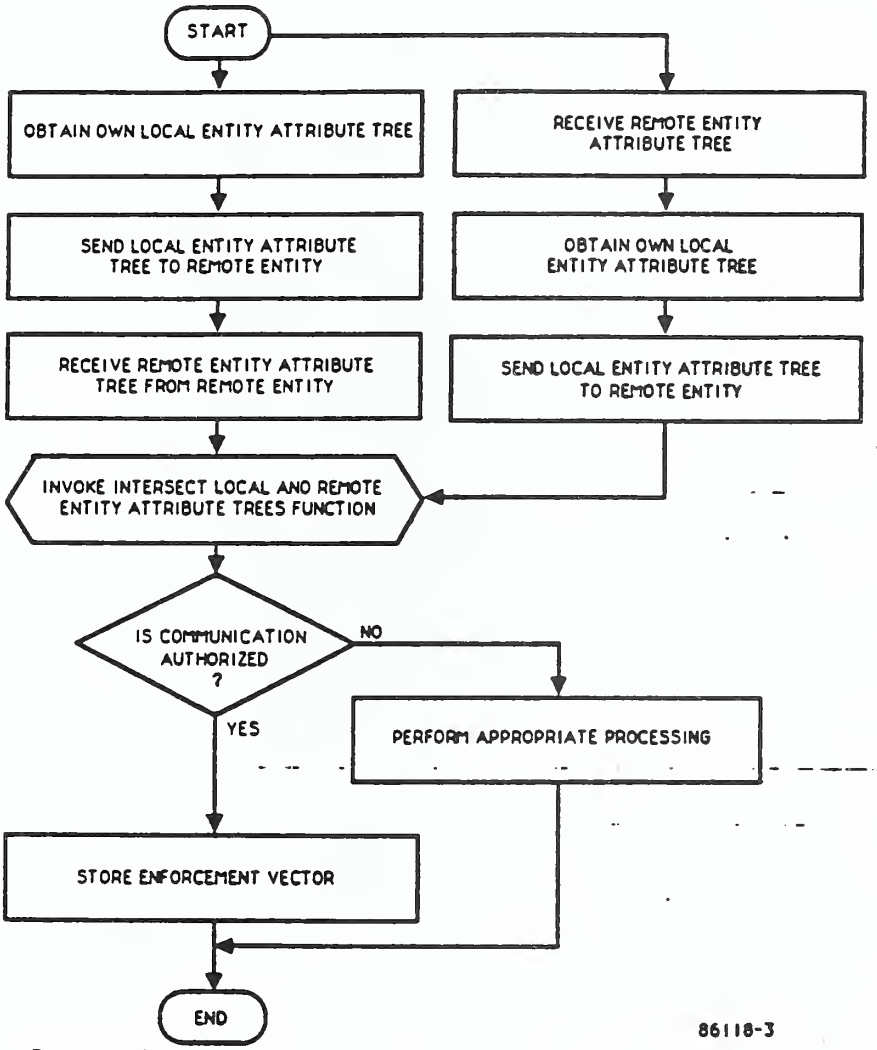


Figure 4-1 Process Performed By Each Entity To Authorize Communication (Process 2)

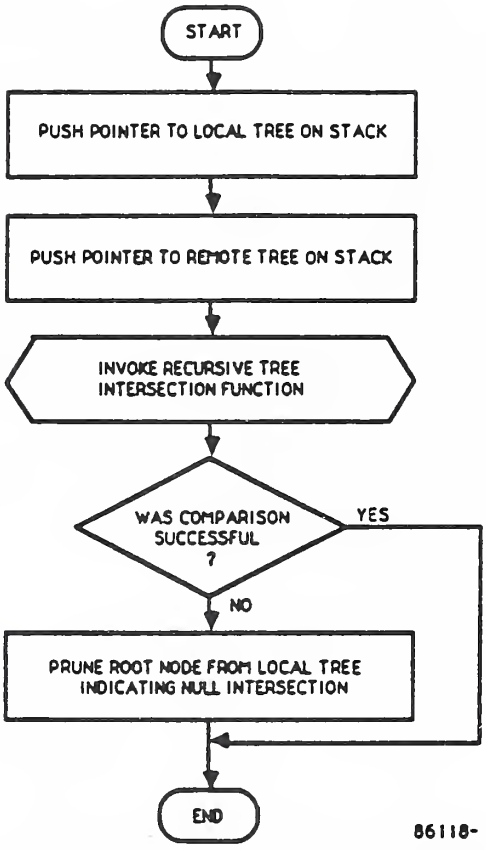


Figure 4-2 Intersect Local and Remote Entity Attribute Trees

Figure 4-2 through 4-11 provide detailed flowcharts of the attribute tree intersection process.

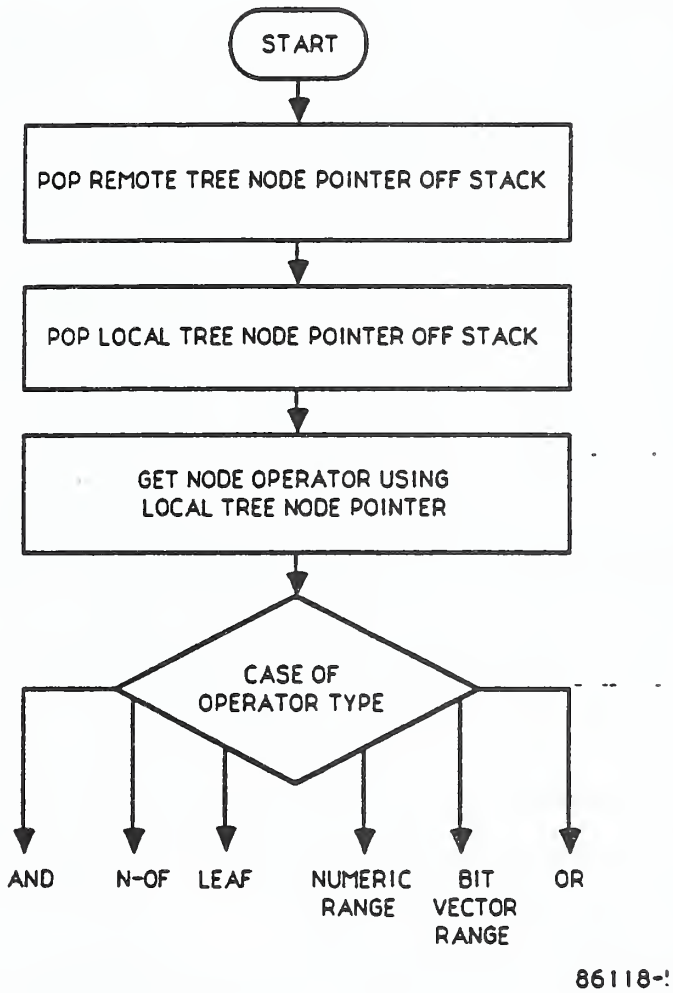


Figure 4-3 Recursive Intersection Testing Function

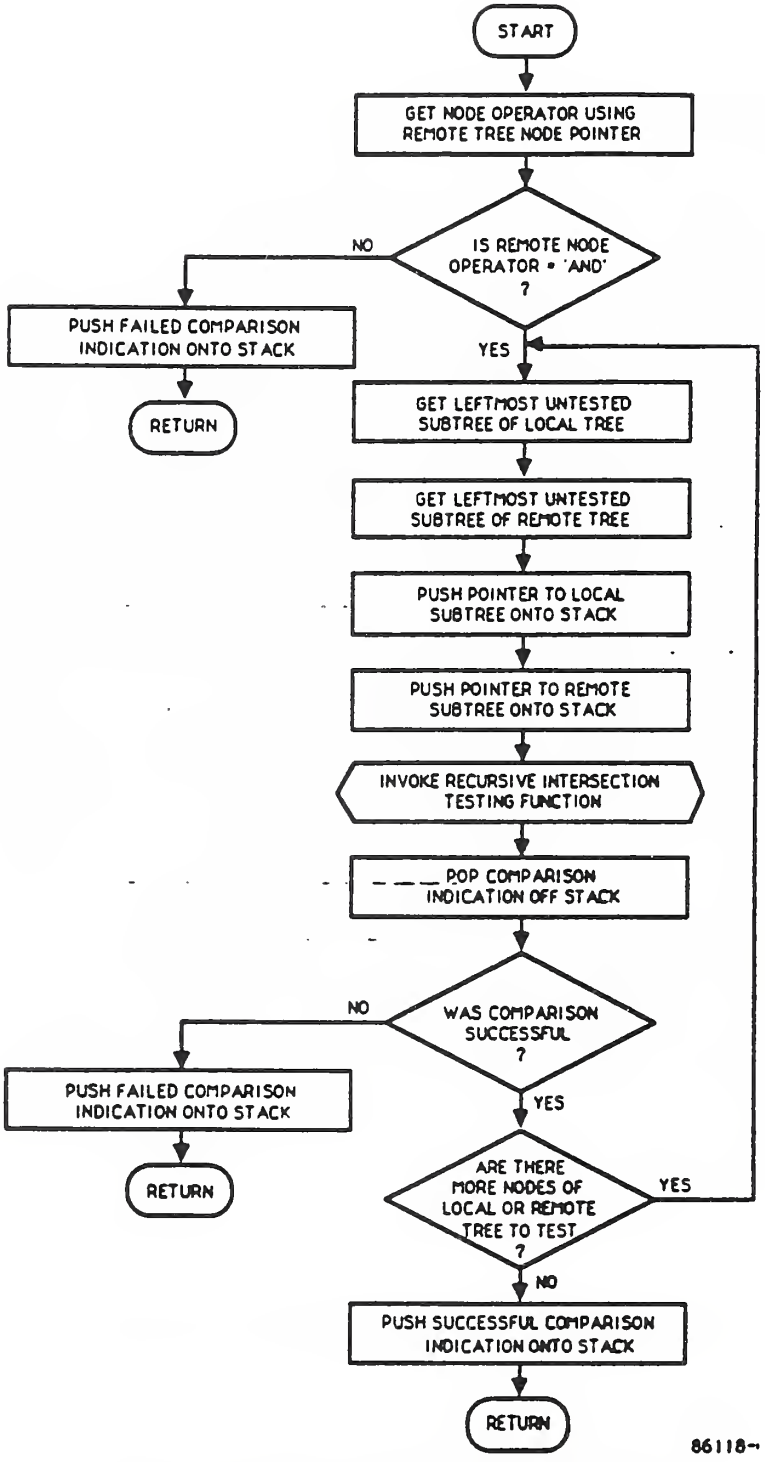


Figure 4-4 "AND" Operator (Intersection)

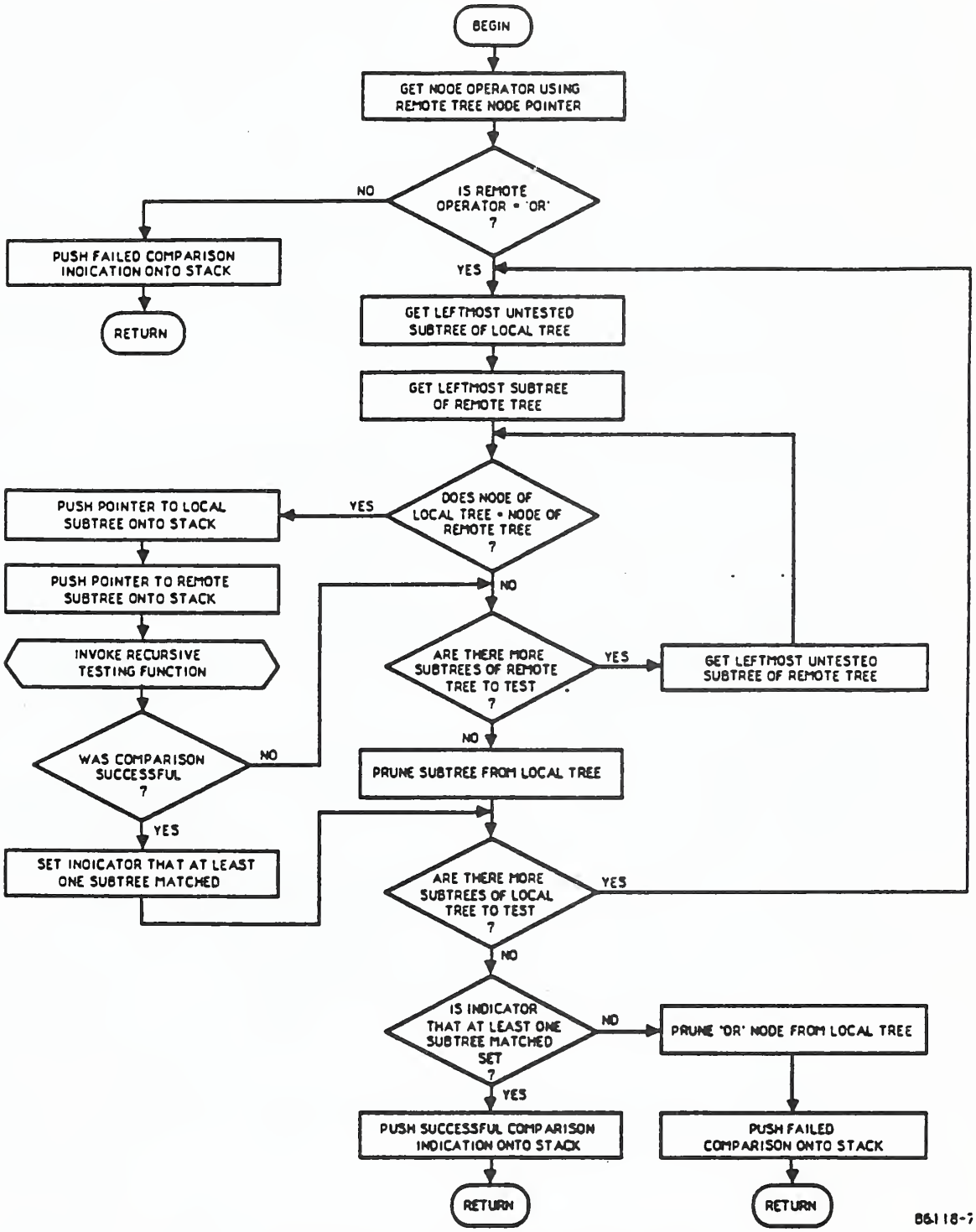


Figure 4-5 "OR" Operator (Intersection)

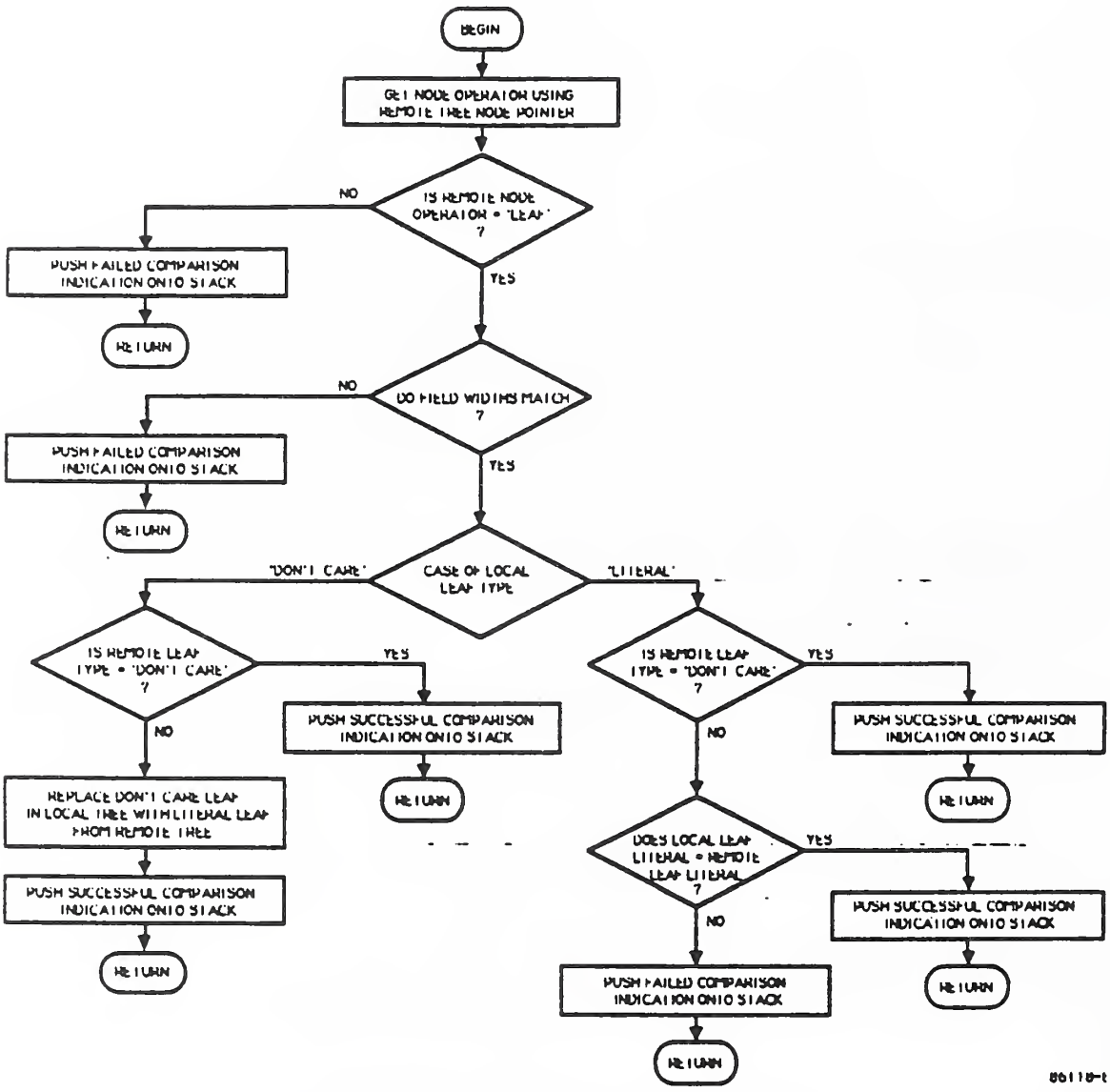


Figure 4-6 Leaf Operator (Intersection)

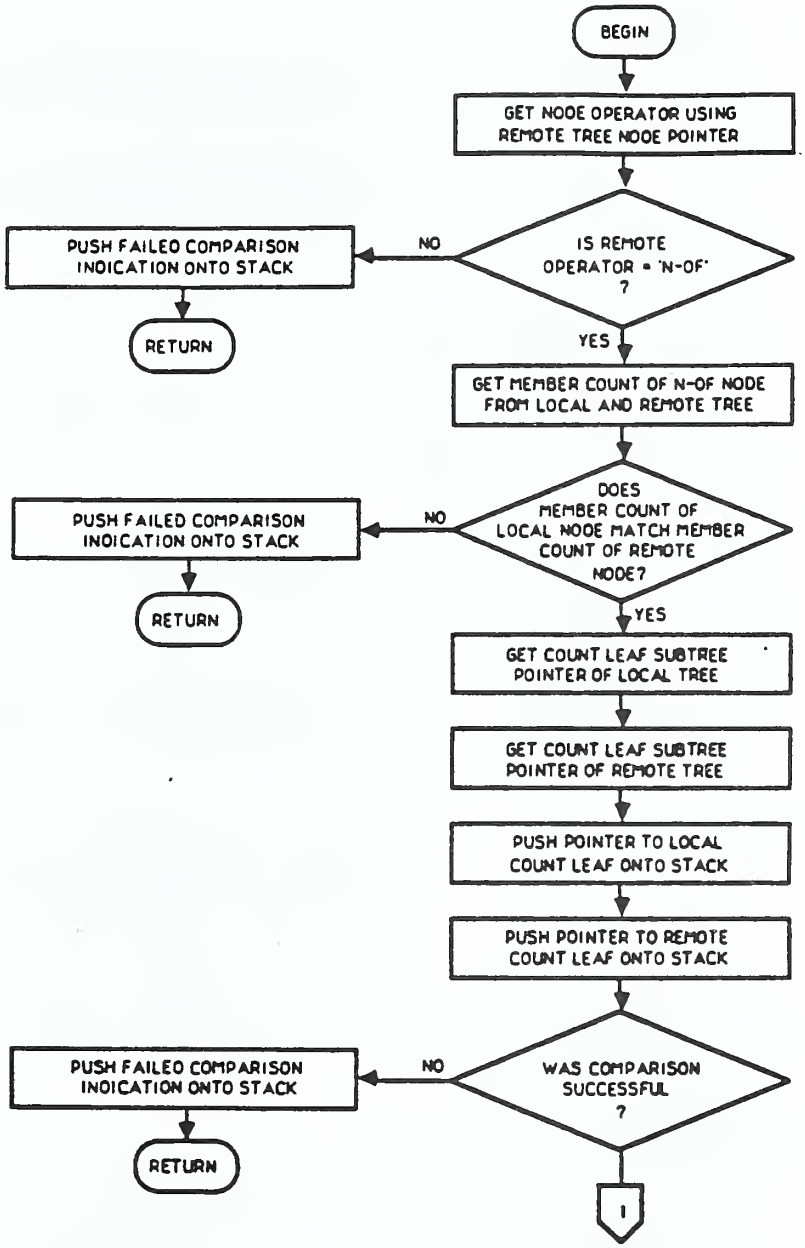


Figure 4-7 "N-OF" Operator (Intersection)

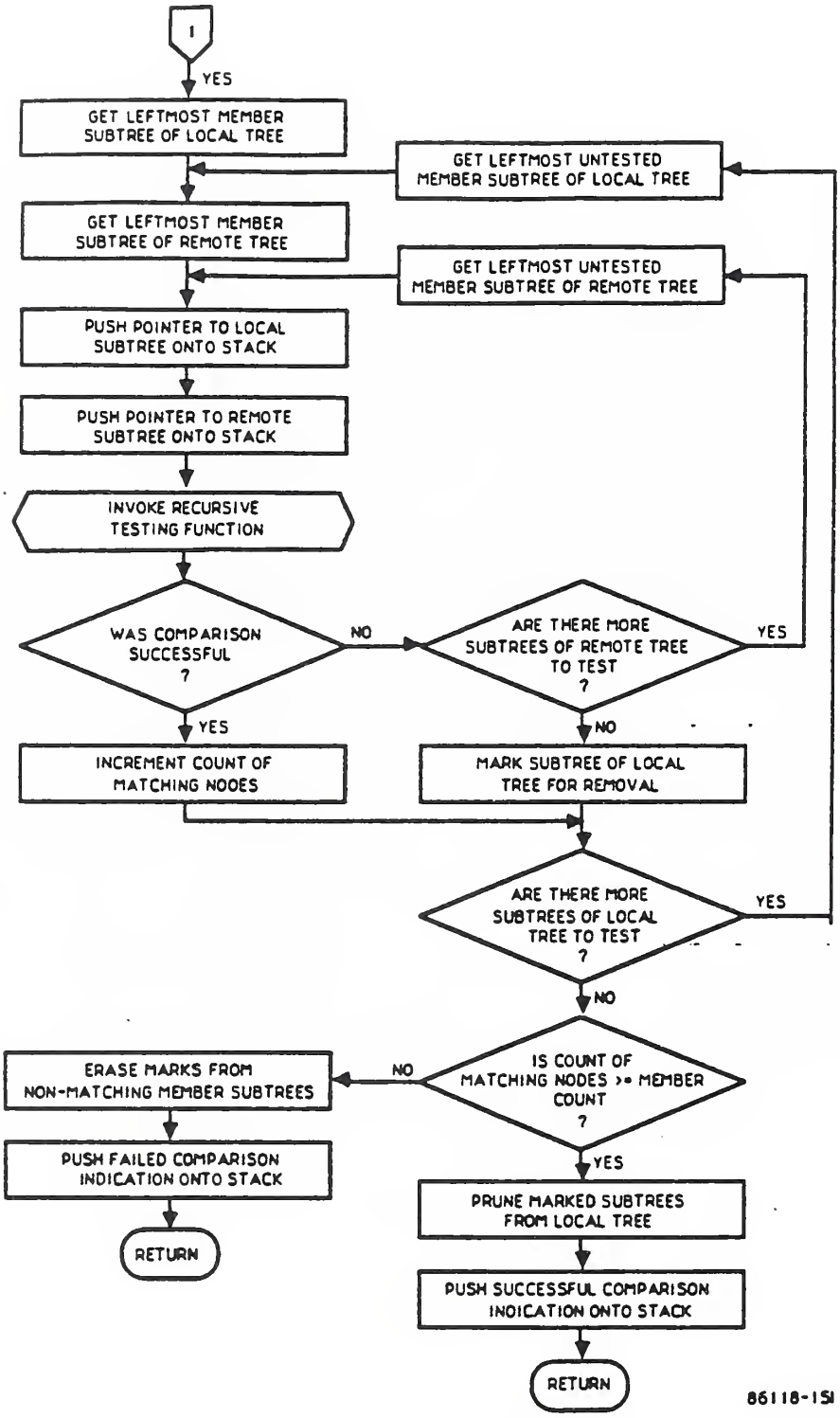


Figure 4-7 "N-OF" Operator (Intersection) (Cont)

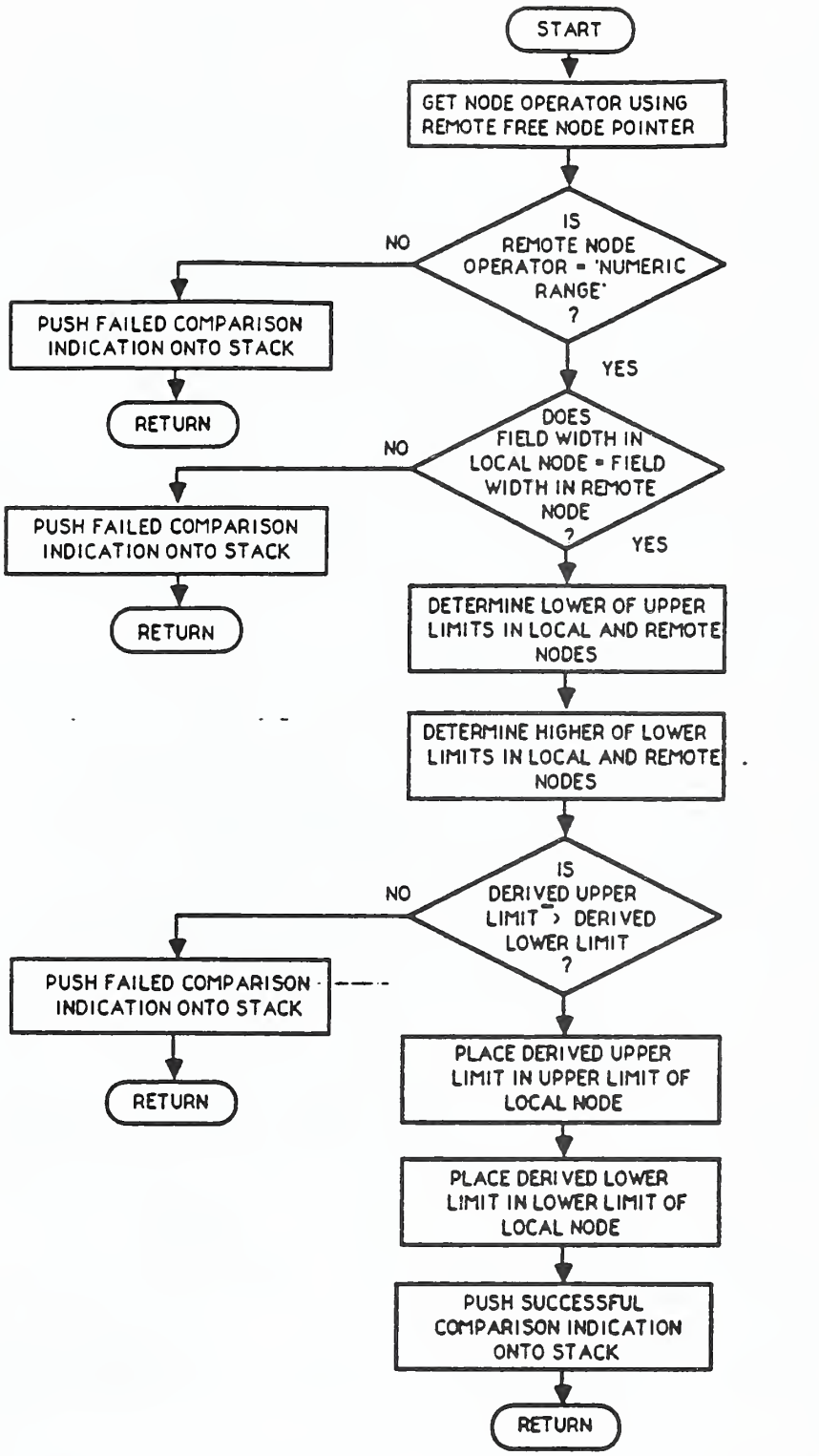


Figure 4-8 "Numeric Range" Operator (Intersection)

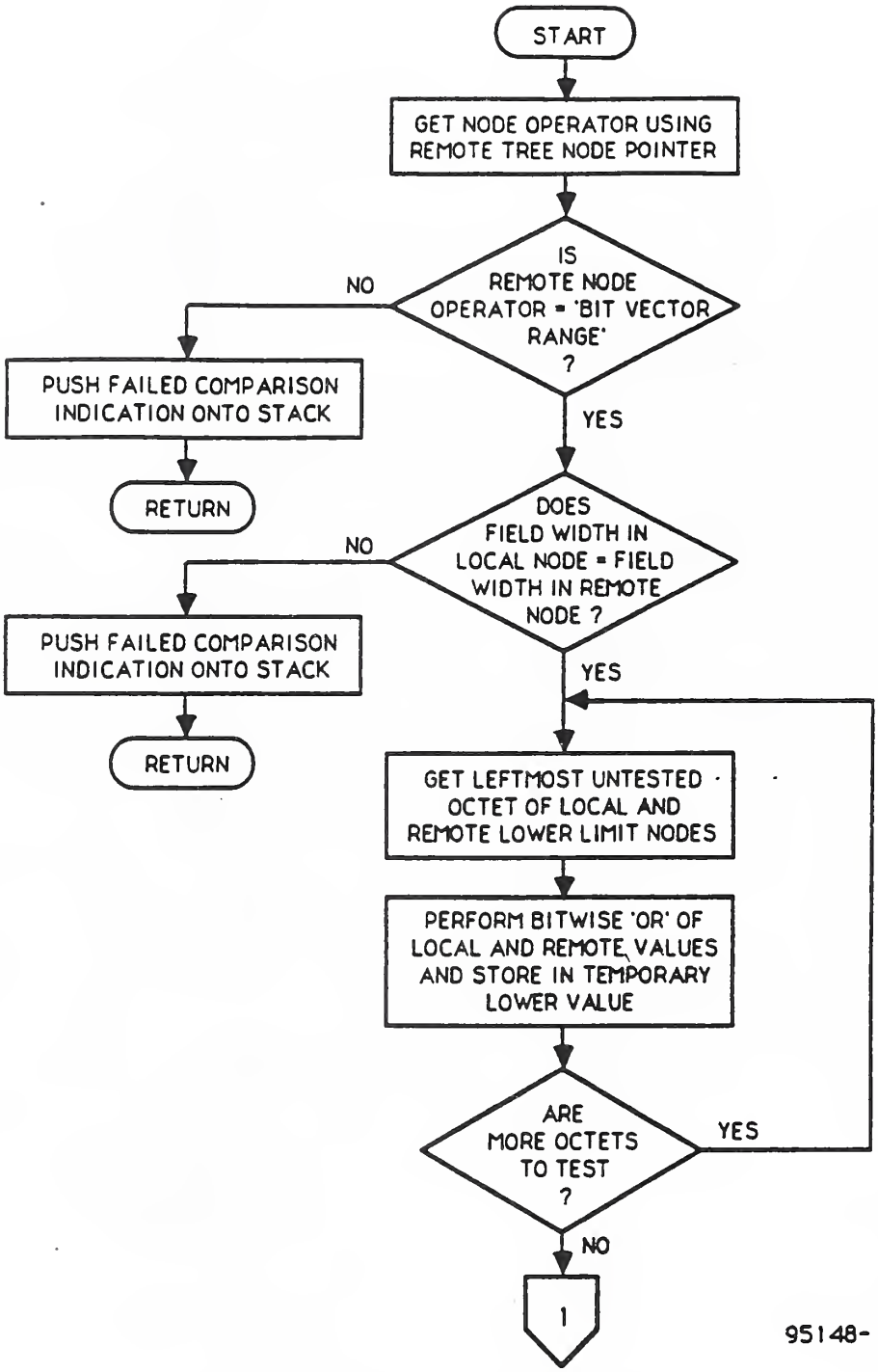


Figure 4-9 "Bit Vector Range" Operator (Intersection)

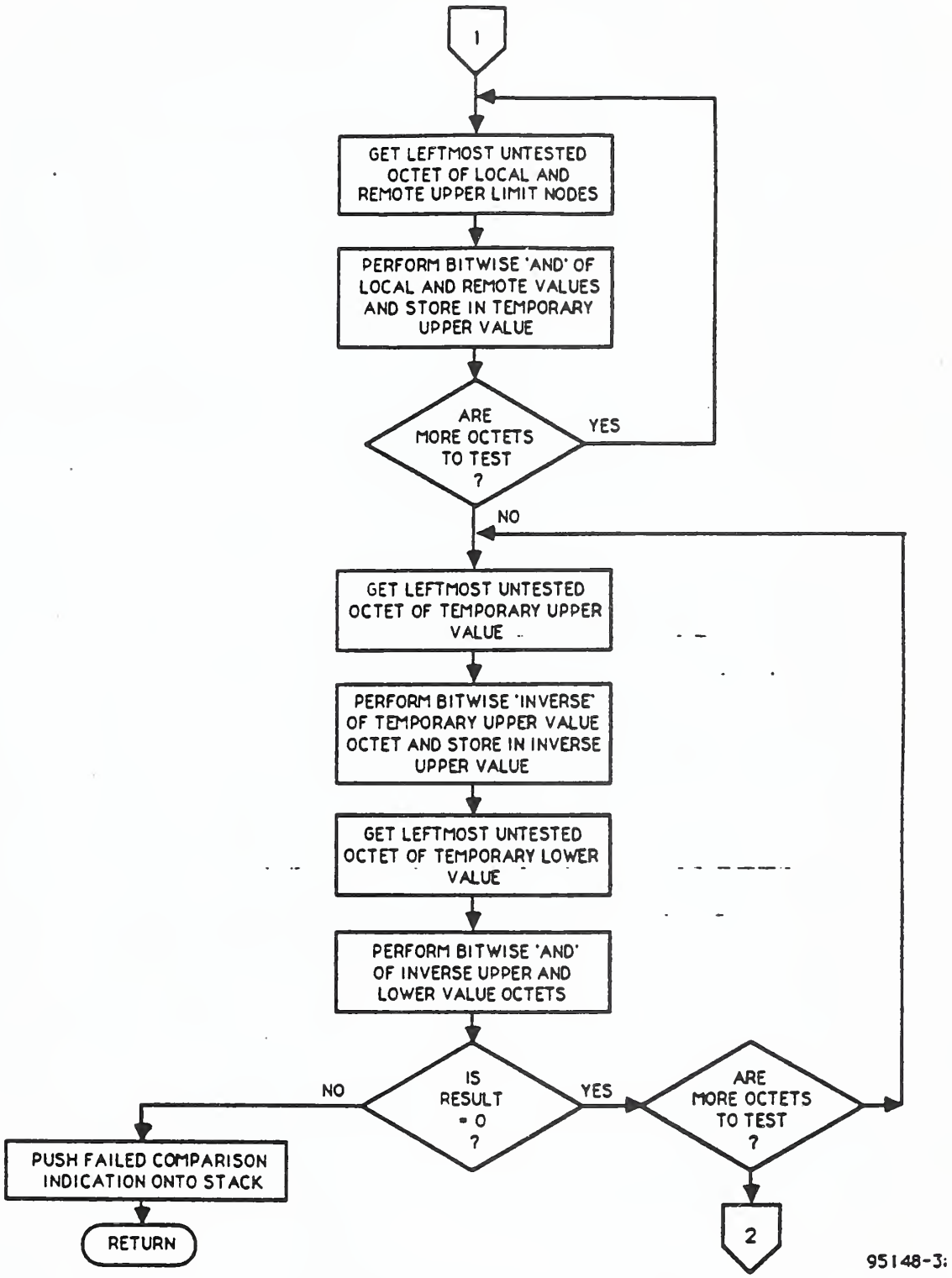


Figure 4-10 "Bit Vector Range" Operator (Intersection - cont)

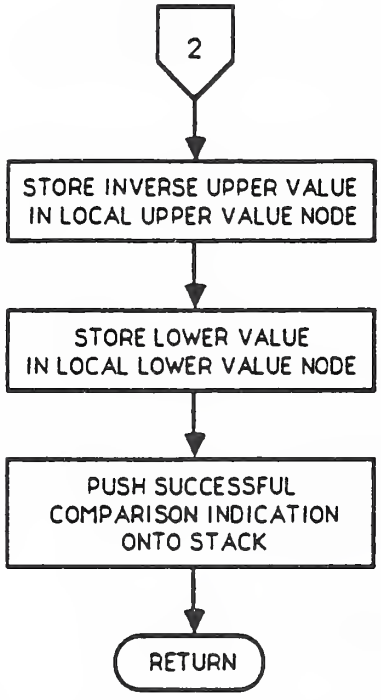


Figure 4-11 "Bit Vector Range" Operator (Intersection - cont2)

5.0 ACCESS CONTROL ENFORCEMENT

Figure 5-1 shows a flow chart of the PDU security label testing process. To begin the process, a determination of the correct EV to use for a PDU is made. The selection criteria may include such items as destination(or source) address and whether an EV has been created per sensitivity label. If the appropriate EV does not exist for some reason (e.g., a connection has not yet been established) then appropriate error handling must be performed.

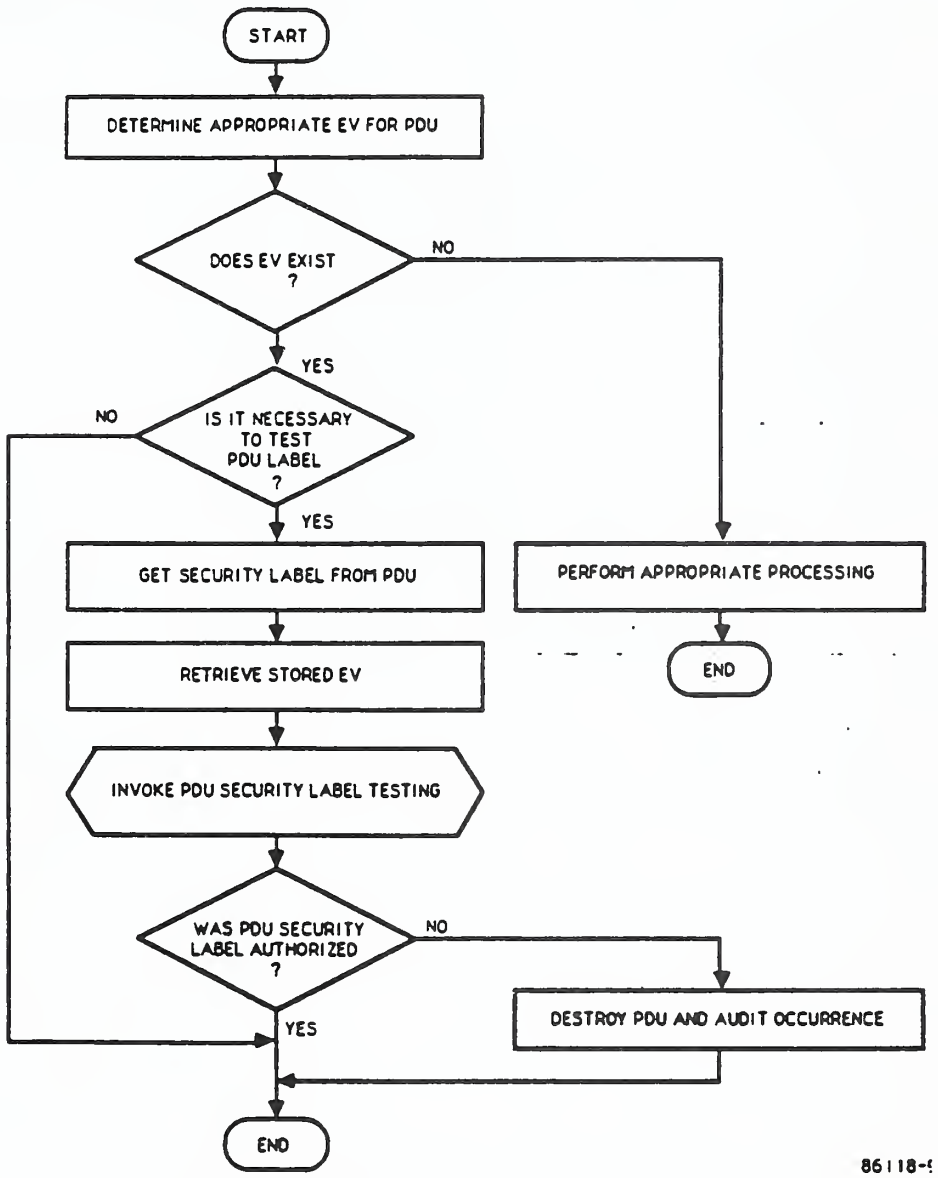


Figure 5-1 Test PDU Security Label (Process 3)

Next a decision is made whether a PDU label test is necessary. This is done because not all PDUs need to contain PDU security labels (e.g., single-level hosts need not label). If a label test is not required, the RBAC portion of PAE is complete.

If label testing is required, then the security label of the PDU (actually a pointer to the security label and the EV generated during PAA are retrieved (actually a pointer to the EV). The security label testing function is then invoked with the security label and the EV. The PDU security label testing function which is shown in Figures 5-2 through 5-10 then determine if the label matches the labelling requirements expressed in the EV. If the label passes the label testing then the PDU is authorized with respect to the ACIS RBAC rules and this part of the PAE process is complete. If the label is unacceptable then appropriate error handling of the PDU is required. This error handling may include destroying the PDU and auditing the event.

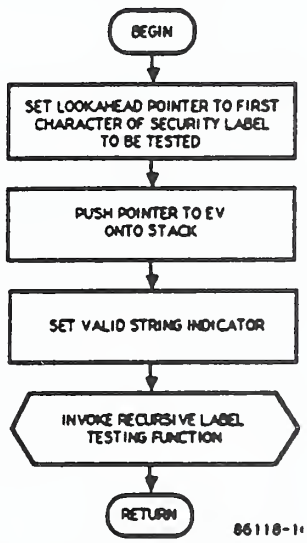


Figure 5-2 PDU Security Label Testing

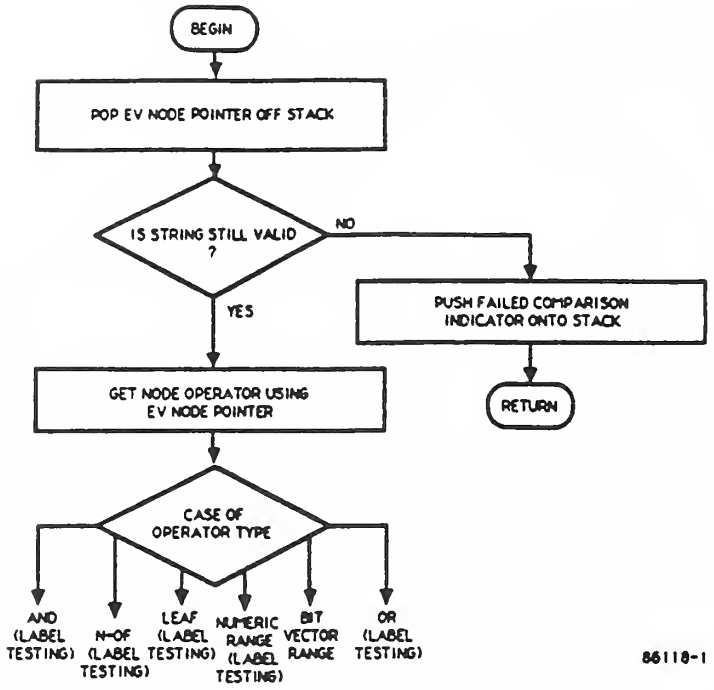


Figure 5-3 Recursive Label Testing Function

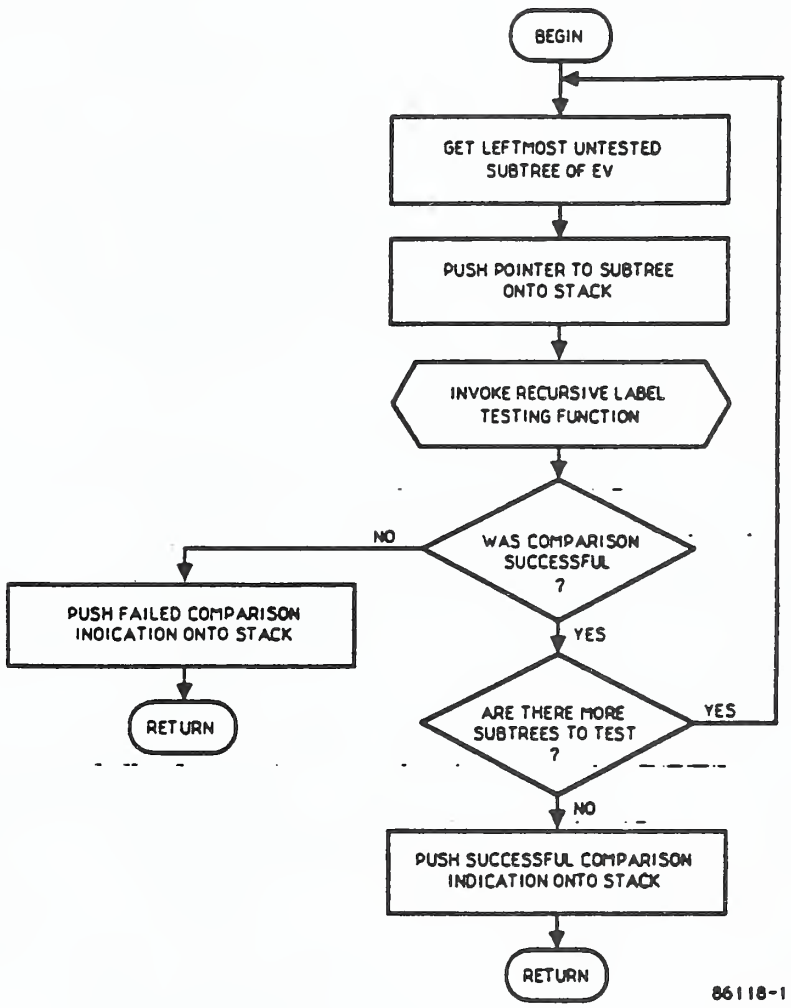


Figure 5-4 "AND" Operator (Label Testing)

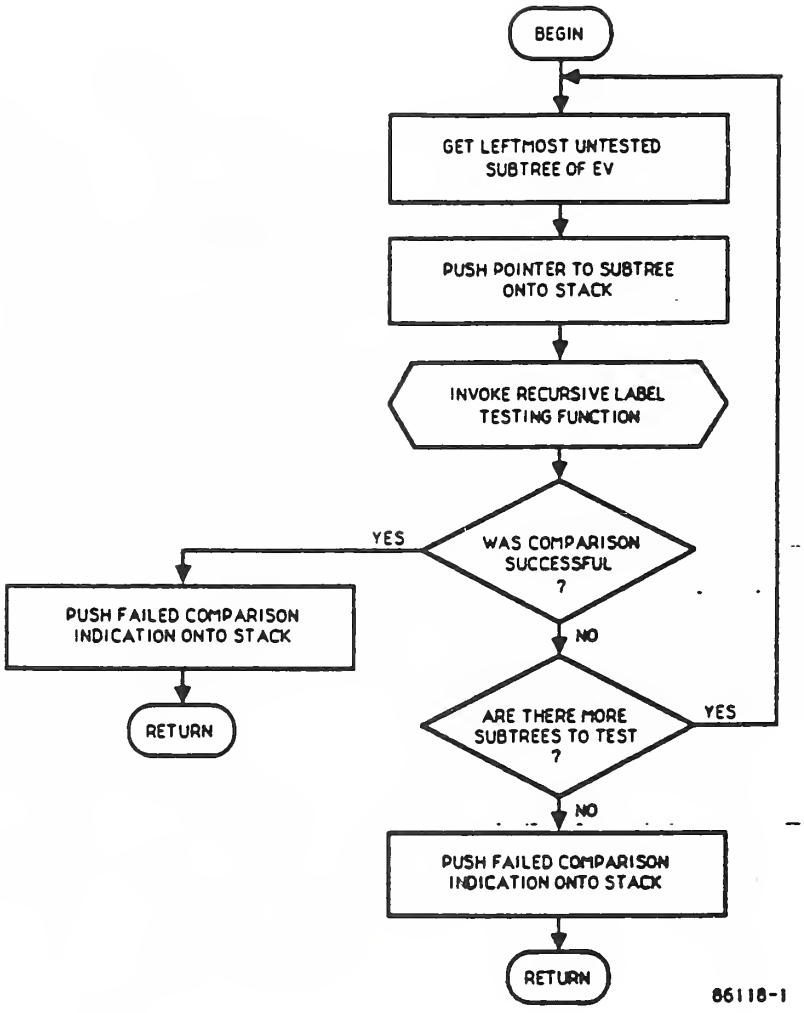


Figure 5-5 "OR" Operator (Label Testing)

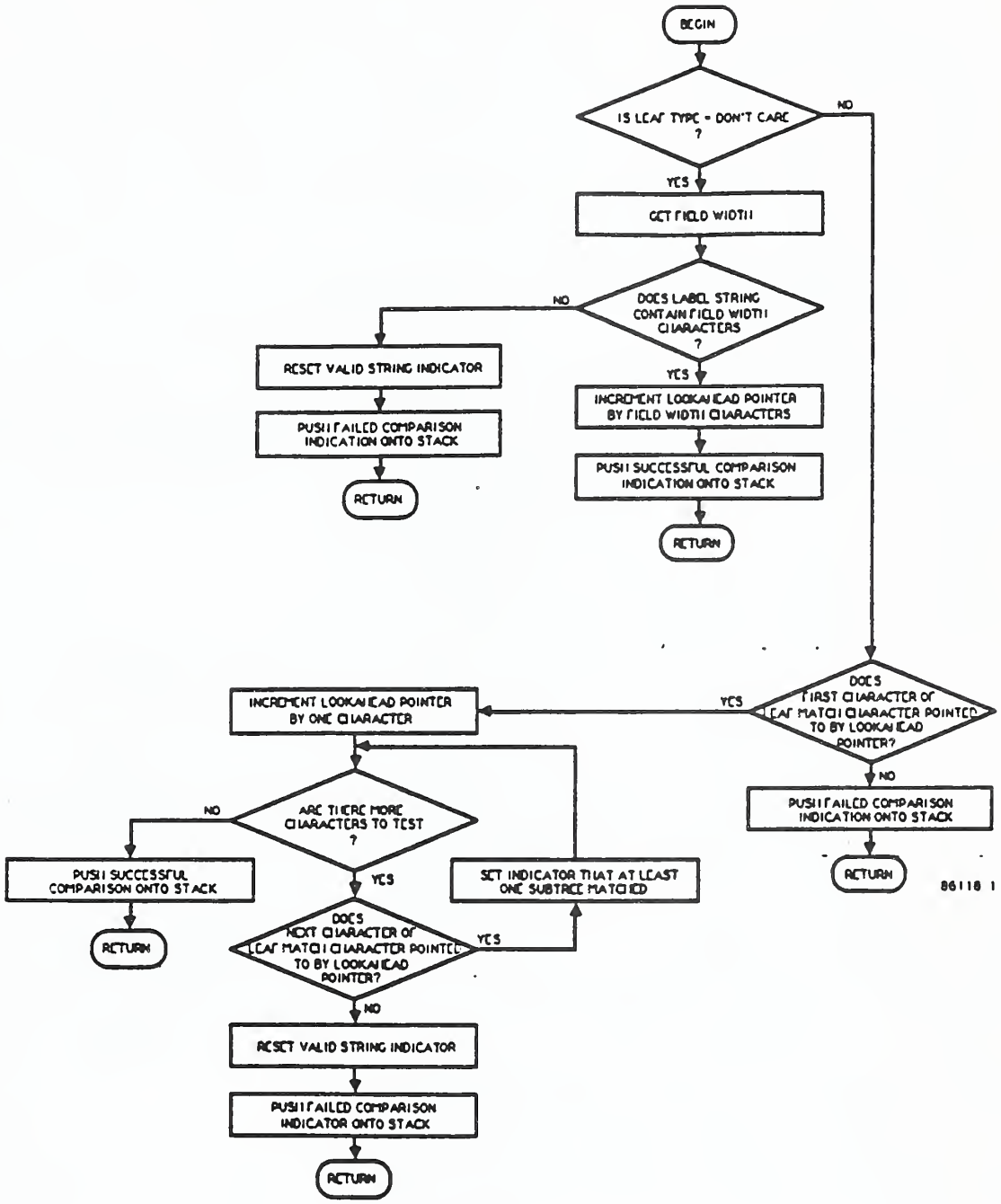


Figure 5-6 "Leaf" Operator (Label Testing)

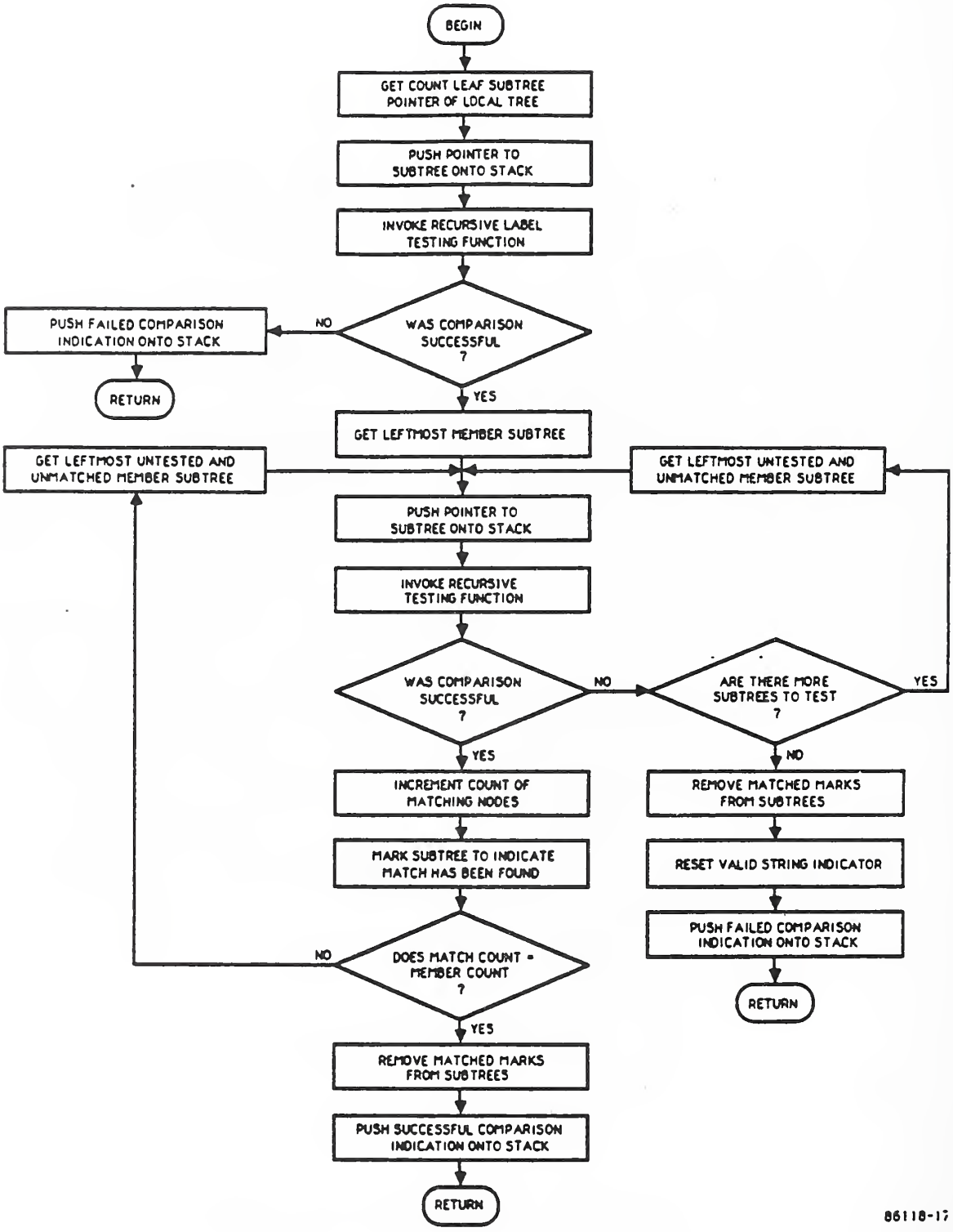


Figure 5-7 "N-OF" Operator (Label Testing)

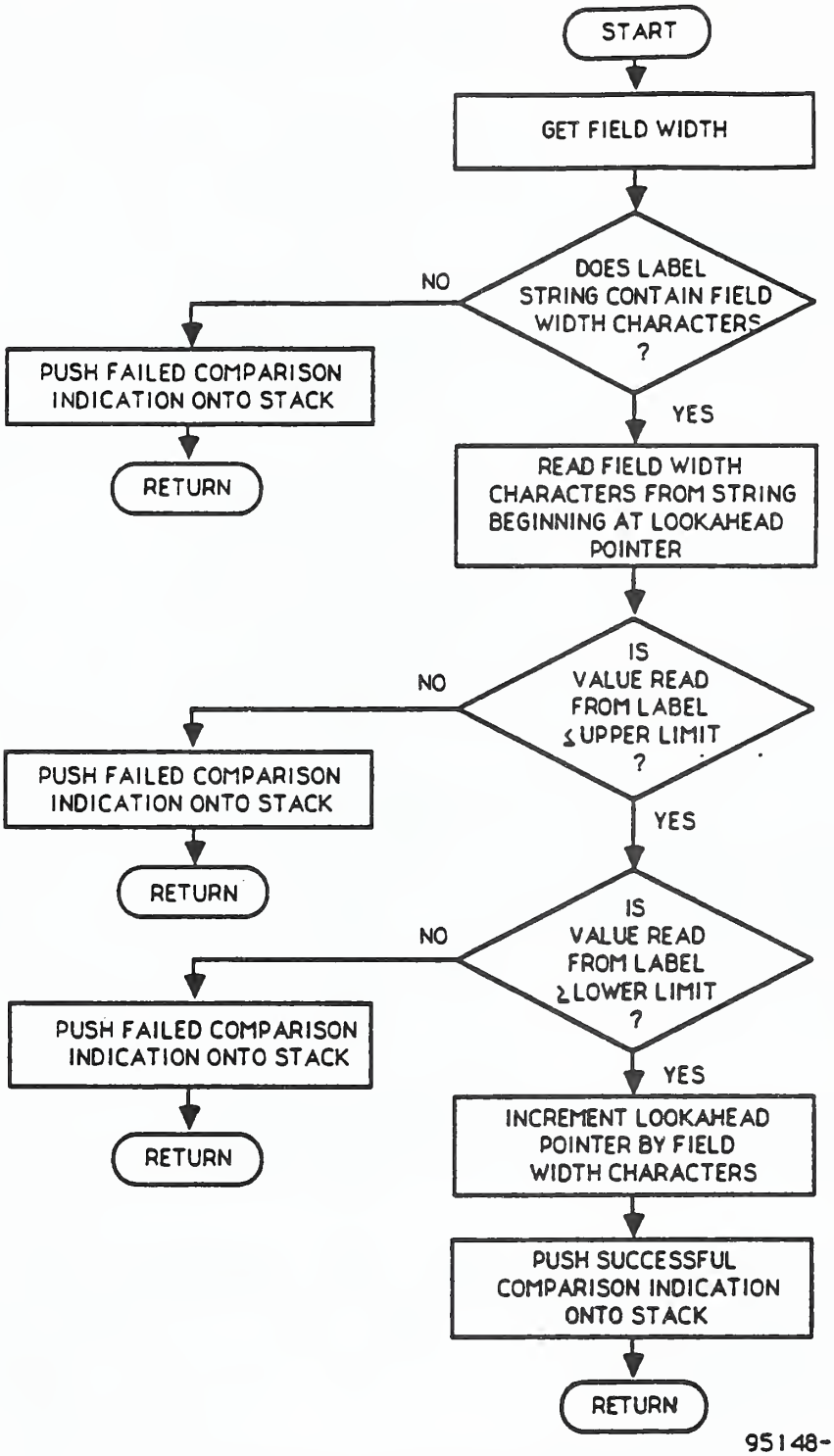


Figure 5-8 "Numeric Range" Operator (Label Testing)

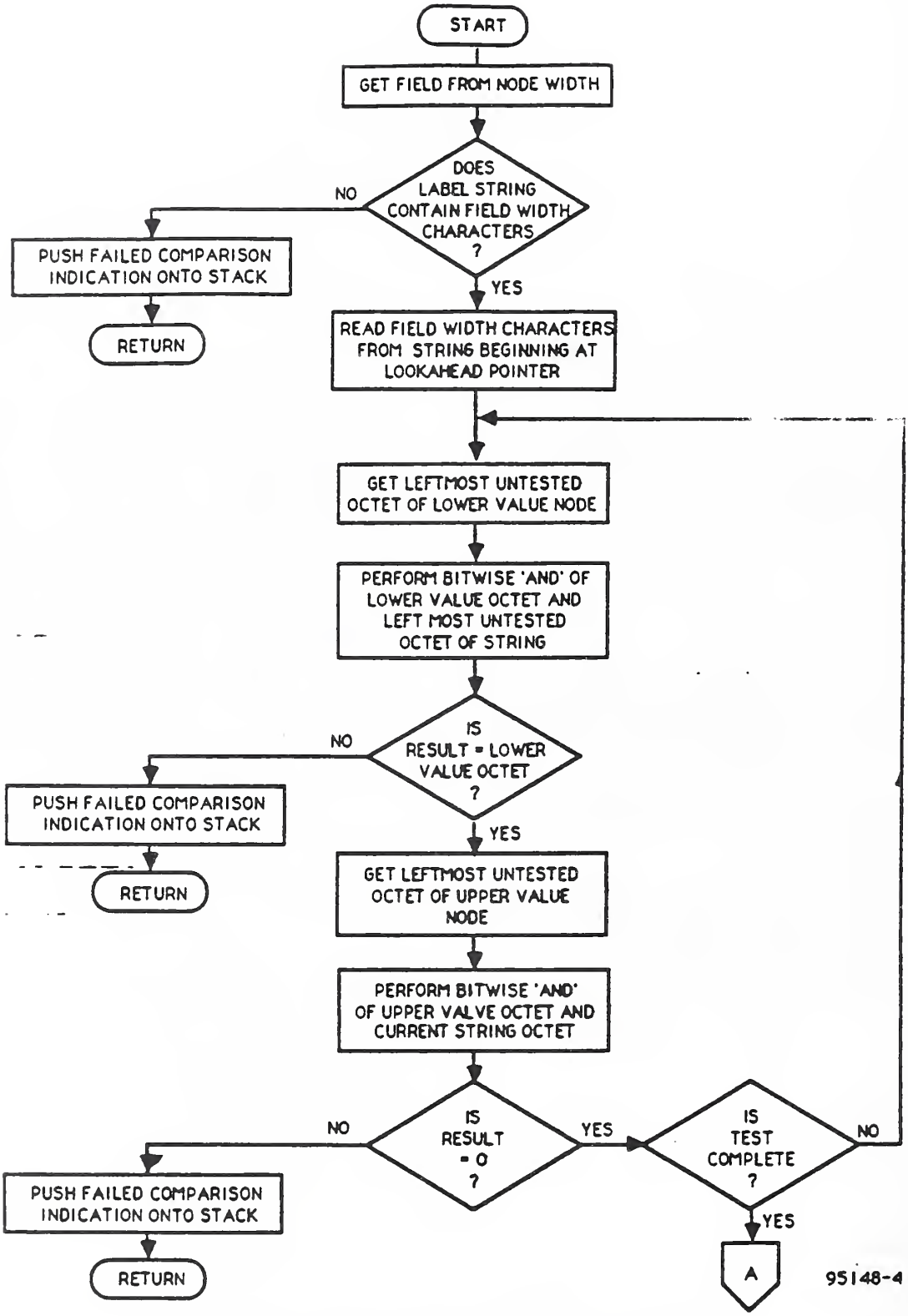


Figure 5-9 "Bit Vector Range" Operator (Label Testing)

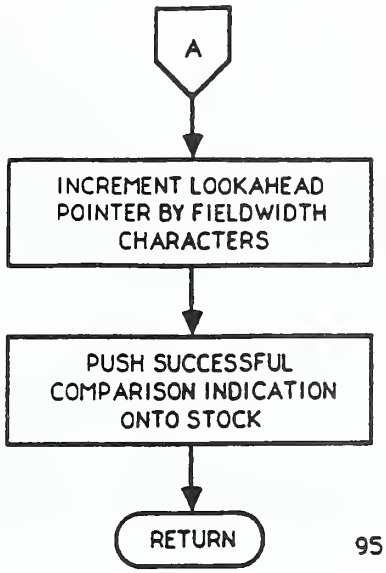


Figure 5-10 "Bit Vector Range" Operator (Label Testing) (Cont)

6.0 GENERAL COMPILER PRINCIPLES

This goal of this review is to lay a foundation for the concepts and terminology used earlier in the ACIS grammar definition in Section 2.2. I am particularly indebted to Alfred V. Aho, Ravi Sethi, and Jerry D. Ullman[2] from whose work, Compilers, Principles, Techniques, and Tools, this summary is derived. The interested reader is strongly encouraged to review the original for a much more complete discussion of the concepts presented. In general, this summary follows these authors' conventions and terminology unless explicitly noted.

6.1 Grammar Components

A grammar is a natural way of describing the hierarchical structure of programming languages or for defining the interrelationship of various access control attributes in ACIS.

Terminals are the most basic symbols from which sentences in a language are formed. The word "token" is a synonym for "terminal". Tokens represent some meaningful string of characters drawn from the alphabet of characters (e.g., A-Z a-z 0-9 ( ) , ; .) defined for the

---

2 Compilers - Principles, Techniques, and Tools, Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman, Addison-Wesley Publishing Company, 1986.

programming language. They constitute the lowest level elements defined for a language hence, the name terminal. Everything else in the language must be constructed from them. For a programming language compiler a token might be assigned to the character string BEGIN, or the character string ":=" (used as an assignment operator). For ACIS, the terminals define such elements as the actual values to be found in PDU labels or some elemental access control attribute.

Nonterminals are the syntactic variables that are built from the ordered collection of terminals and/or other nonterminals. The nonterminals provide the hierarchical structure to grammars since at the lowest level they are constructed exclusively of terminals but as one moves down the hierarchy they are made of other terminals and/or nonterminals (which in turn could be made up of other nonterminals).

One of the nonterminals is distinguished as the start symbol. Typically this symbol will be implicitly denoted as such by appearing as the first production in the grammar. The set of strings that can be derived from this variable constitute the language defined by the grammar. For ACIS, this represents the total space of allowable access control attributes including labels.

Lastly, a grammar is composed of productions. These specify the manner in which the terminals and nonterminals can be combined to form terminal strings within the language. A production consists of a nonterminal, called the left side of the production, an arrow ($\rightarrow$), and a sequence of tokens and/or nonterminals. This sequence of terminals and/or nonterminals is called the right side of the production. The ϵ is used to indicate that the right side provides a definition for the nonterminal on the left side of the production. It is possible to define a string containing neither terminals nor nonterminals. This string is called the empty string and is denoted by e. For notational convenience, productions with the same nonterminal on the left side can have their right sides grouped with the alternative right sides separated by a | symbol. This symbol is read as an "or".

A grammar derives tokens strings (and ultimately character strings when the tokens are replaced by the literal values they represent) by beginning with the start symbol and repeatedly replacing a nonterminal with the right side of a production for that nonterminal. The replacement process proceeds until terminals are reached. The token strings that can be derived from the start symbol form the language defined by the grammar.

6.2 Parsing

One way to ensure that a particular string is part of a language defined by a grammar is to use the productions of that grammar to derive the string in question. A parse tree is a graphical representation of such a derivation. For the sake of example, assume the following grammar (called grammar1):

1. list → list + digit
2. list → list - digit
3. list → digit
4. digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

or equivalently:

list → list + digit | list - digit | digit
digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

The productions in the first representation have been numbered in order to easily refer to them. The tokens (or terminals) for this grammar are 0 1 2 3 4 5 6 7 8 9 + -. Terminals are identified by the bold font. The nonterminals are list and digit. Digits are defined in terms of the terminals 0 - 9. The nonterminal list is defined in terms of the nonterminals list and digit and the terminals + and -. List is the starting nonterminal because its production is given first.

If we are given the string 9 - 5 + 2, we can test to see if this conforms to the grammar by attempting to build a parse tree for it. If one can be built, the string is acceptable, otherwise it is not. Figure 6.2-1 shows such a parse tree.

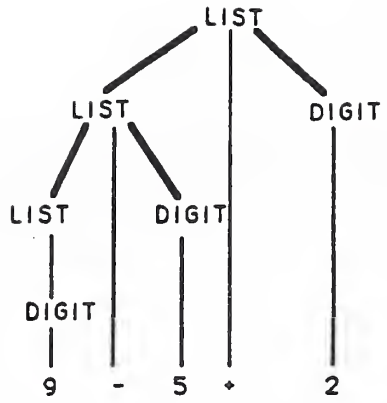


Figure 6.2-1 Parse Tree From Grammar1 for Example String

a. 9 is a digit by production 4.
b. 9 - 5 is a list by production 1, since 9 is a list by production 3 and 5 is a digit by production 4.
c. 9 - 5 + 2 is a list, since 9 - 5 is a list and 2 is a digit. Since it was possible to construct a parse tree for the string 9 - 5 + 2, the string is shown to be a valid string in the language defined by grammar1.

A parse tree has the following properties:

1. The root is labeled by the start symbol.
2. Each leaf is labeled by a token or by e.
3. Each interior node of the tree is labeled by a nonterminal.
4. If A is the nonterminal labeling some interior node and $X_1, X_2, \ldots X_n$ are the labels of the children of that node from left to right then $A \rightarrow X_1 X_2 \ldots X_n$ is a production. Here $X_1, X_2, \ldots X_n$ stands for a symbol that is either a terminal or a nonterminal. As a special case, if $A \rightarrow e$ then a node labeled A may have a single child labeled e.

6.3 *Recursion*

As can be seen from the grammar1 example, general grammars can contain productions which include recursive definitions of nonterminals (i.e., the nonterminal appearing on the left side of a production also appears on the right side of the production. Productions can exhibit either right or left recursion depending upon where the nonterminal being defined appears in the right hand side of the production. Left-recursion occurs when the nonterminal being defined also appears as the first (or leftmost) symbol on the right side of the production. For example consider the

production A → Aa | b. The parse tree for the string baaaaa is shown in Figure 6.3-1. This production is left recursive because the nonterminal A appears as the leftmost symbol in the definition of the nonterminal A.

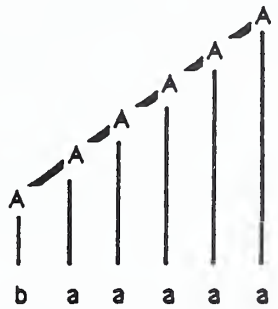


Figure 6.3-1 Parse Tree Derived From Left-Recursive Production

Similarly a production can exhibit right recursion if the nonterminal on the left side of the production also appears as the rightmost symbol in a production. Let A be defined by the following grammar:

A → bR
R → aR | e

A new nonterminal R has been defined. It is right-recursive per the definition given above. Figure 6.3-2 shows the parse tree associated with the string baaaaa using the right-recursive grammar. Recursive productions are not allowed in ACIS grammars.

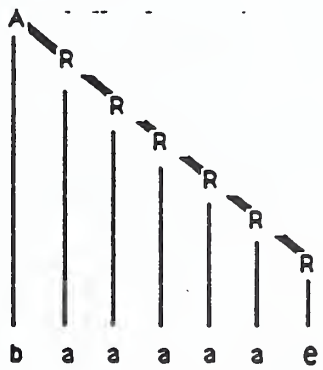


Figure 6.3-2 Parse Tree From Right-Recursive Grammar

6.4 *Top-Down Parsing*

There are basically two kinds of parsing algorithms. They are top-down or bottom-up. These refer to the way in which nodes in the parse

tree are constructed. In the top-down methods, construction begins with the root using the starting nonterminal and proceeding towards the leaves (i.e., down toward the terminals or tokens). In bottom up parsing, construction begins with the leaves and proceeds upward toward the root (i.e., up toward the starting nonterminal).

The top-down construction of a parse tree is done by starting with the root, labelled with the starting nonterminal, and repeatedly performing the following two steps:

1. At node n, labelled with nonterminal A, select one of the productions for A and construct children at n for the symbols on the right side of the production.
2. Find the next node at which a subtree is to be constructed. This process is continued until tokens have been reached at each of the subtrees. The tokens in the parse tree correspond to the tokens in the token string being tested.

Given the following grammar (grammar2) and the token string b x c, the parse tree shown in Figure 6.4-1 can be constructed using the algorithm given above.

type → A | a | b A c
A → x | y

The root of the parse tree is constructed using the starting nonterminal, type. This is shown in the first part of Figure 6.4-1. Since type is a nonterminal, it is necessary per step 1 of the algorithm to select one of the productions for type and construct children corresponding to each of the symbols on the right side of that production. In this example there are three possible choices that could be made for the definition of type. It could be defined by the nonterminal A, by the terminal a, or the terminal b followed by the nonterminal A followed by the terminal c. In order to determine which production should be selected, the terminal string being tested is used. This is done by scanning the token string in a left to right manner. The vertical arrow will be used to indicate the next character to be scanned and is called the lookahead symbol. In this example, it is clear that the production, type → b A c, is the correct choice because it is the only production that allows the token string to begin with the token b.

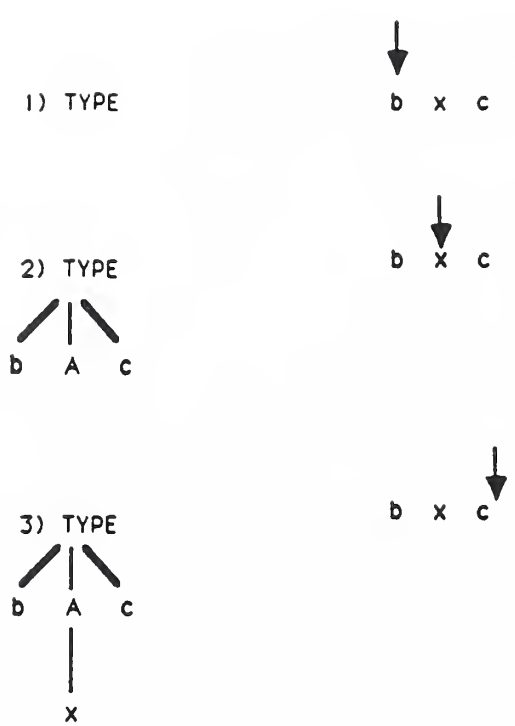


Figure 6.4-1 Top-Down Construction Of Parse Tree For String bxc

After the production is selected, the children are constructed for this production and the lookahead symbol is advanced to the next token, x. This symbol will be used to determine which production is to be used for the nonterminal A. These actions are indicated in part 2 of the figure.

Next the children just constructed need to be tested to see if they need further definition. The first child, b, is a terminal so its definition is complete. The second symbol, A, however is a nonterminal so it is necessary to select a production to provide further definition. The nonterminal A can be defined by either the terminal x or y. Using the lookahead symbol, x, the first definition is selected. Construction of this child is shown in part 3 of the figure. Because x is a terminal, no more children are constructed at that node. After using the symbol x in the token string, the lookahead symbol is advanced to the symbol c. This will be used to test the third child in the definition of type. Since the symbol in the parse tree and the symbol in the token string match (i.e., they are both c) and because this is the last child, the parse tree is complete. The lookahead symbol was advanced beyond the c to ensure that the token string contained no other characters to be parsed. Notice that the token string to be parsed can be found at the leaves of the parse tree. Since a parse tree could be generated for the token string, bxc is shown to be a

valid string in the language defined by grammar2. If it had been impossible to generate such a parse tree, then the token string would have been declared invalid.

In the example just presented it was straightforward to determine which production to select when constructing the parse tree using the lookahead symbol. For example, the only variable production for type was type → b A c. This was because it is the only production that allows for a token string to begin with b. If our grammar had also included the production type → b A d. Either production could have been selected when using only the lookahead symbol b. Each production will also work with the second symbol x. It is not until the lookahead symbol is advanced to the third (in this case c) that the correct choice can be determined. Thus if we had chosen the second production (type → b A d), it isn't until we test the third child d that we determine that this does not match the token string under test. At this point in order to pick the alternate production it is necessary to reset the lookahead symbol to b and to replace the incorrect node definition of type with the correct one, type b A c.

In general, then the production selection process may involve trial and error. If a production is selected which does not allow the successful definition of a parse tree, it is necessary to backtrack and select an alternate production until either a parse tree has been built or all possibilities have been exhausted. There is a special class of grammars, however, in which backtracking is not required. These grammars have the characteristic that they do not allow ambiguities to exist with respect to the choice of production to choose when constructing a parse tree. At any given decision point, only one possible choice exists. These grammars are called predictive parsers. They are especially suited to the hand generation of grammars using the top-down definition illustrated above and is the kind of grammar recommended for ACIS.

NIST-114A
(REV. 3-89)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 90-4259

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
MARCH 1990

4. TITLE AND SUBTITLE

5. AUTHOR(S)

Charles Dinkel (Editor)

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

National Security Agency - Information Security Applications Group
9800 Savage Road, SDNS SP0 (C23)
Ft. Meade, MD 20755-6000

10. SUPPLEMENTARY NOTES

[ ] DOCUMENT DESCRIBES A COMPUTER PROGRAM; SF-185, FIPS SOFTWARE SUMMARY, IS ATTACHED.

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

The Secure Data Network System project, known as SDNS, implements computer to computer communications security for distributed applications. The internationally accepted Open Systems Interconnection (OSI) computer networking architecture provides the framework for SDNS. SDNS uses the layering principles of OSI to implement secure data transfers between computer nodes of local area and wide area networks. This publication includes three access control documents developed by the National Security Agency (NSA) as output from the SDNS project. SDN.801 describes the principles and functions underlying the SDNS access control and authentication security services. SDN.802 provides specifications which form a common basis from which devices implementing the access control service will be able to achieve interoperability. The third document in the set, SDN.801 Addendum 1, is an extension of access control information specifications provided in section 5 of SDN.802.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

access control; authentication; computer security; network security; SDNS; security protocols

13. AVAILABILITY

[X] UNLIMITED

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.

[X] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
161

15. PRICE
A08

ELECTRONIC FORM